UNIVERSIDAD CARLOS III DE MADRID

Escuela Politécnica Superior

INGENIERÍA TECNICA EN INFORMÁTICA DE GESTION

PROYECTO FIN DE CARRERA

IMPLANTACIÓN DE UN ERP (MICROSOFT DYNAMICS NAV:NAVISION) EN UNA EMPRESA DE JUGUETES

AUTOR: Miguel Angel Carbonero Jiménez
TUTOR: Miguel Angel Ramos [08/09/2011]

Índice de Contenidos

Abreviaturas y Acrónimos

MRP	*Material Requirement Planning*
CRP	*Capacity Requirement Planning*
ERP	*EnterpriseResourcing Planning*
TPV	*Terminal de Punto de Venta*
ADSL	*Asymmetrical Digital Subscriber Line*
SHDSL	*Single pair High-speed Digital Subscriber Line*
VPN	*Virtual Private Network*
LAN	*Local Area Network*
PDA	*Personal Digial Assistant*
FIFO	*First In First Out*
FEFO	*First Expiry First Out*
LIFO	*Last In First Out*
XML	e*Xtensible Markup Language*

Índice de Ilustraciones

1 Introducción

La gestión de almacén dentro de la pequeña y mediana empresa ha ido evolucionando a lo largo de los años, buscando una gestión que sea lo más automatizada posible, ya que se intenta que la intervención humana en las decisiones de dicha gestión sea lo menos relevante posible. Se pretende que dichas decisiones puedan contribuir de forma positiva a que la gestión del almacén sea lo más eficaz posible. Por ello, se explicará a continuación, como ha ido evolucionando la forma de gestionar un almacén.

En los años 60, no existía ningún sistema genérico de gestión de almacén, todo se hacía en base a la experiencia y suposiciones de los expertos en ese tipo de productos. Pero se comprobó que en un porcentaje alto de las ocasiones, el suministro de las piezas para el montaje de sus productos no funcionaba cómo se esperaba. Esto provocaba que en muchas ocasiones no se pudiera ensamblar el producto final por falta de algún componente.

La solución que se encontró a este problema fue la definición de lista de materiales, que se trata de una estructura jerarquizada análoga al padre-hijo que recoge el árbol genealógico. Sin embargo, dicha lista presentaba numerosas limitaciones. Una de las más importantes era que no calculaba las fechas en las que debían realizarse los pedidos cuando el stock de las piezas que componía el producto estaba por debajo del mínimo para construir una unidad del producto.

Para paliar estas deficiencias se desarrolló el MRP que proyecta en el tiempo las necesidades de materiales. Gracias al MRP, gestionando los productos según la previsión del artículo principal, se puede saber con certeza la demanda de las piezas de las que se compone el artículo, gracias a la lista de materiales. Conociendo los plazos de entrega de los proveedores, resulta trivial determinar cuándo hay que realizar el lanzamiento de los pedidos. El beneficio principal derivado del MRP era una considerable reducción del inventario. Por lo tanto, la reducción del inmovilizado ofreció a las empresas importantes ahorros económicos al obtener beneficio financiero.

Pero con el paso de los años se le exigían a los MRP funcionalidades más altas y pronto se comenzó a solicitar que realizaran la planificación en función de la capacidad de la planta, algo que con los MRP era inviable. Entonces, los resultados que se obtenían con el MRP eran introducidos cómo datos de entrada en el programa llamado

CRP, los cuales tenían en cuenta las restricciones de capacidad, determinaban si la planificación era posible o no.

En caso de resultar inviable se volvían a calcular nuevos lanzamientos con el MRP y se obtenían las recomendaciones con el CRP.

Debido a que era muy tedioso el estar utilizando dos programas, se decidió incorporar al MRP el módulo CRP. Además se aprovechó para incluir módulos de gestión de compras, ventas, almacenes y la contabilidad. Este nuevo modelo de sistema de información se conoció con el nombre de MRP II

Los problemas de final de siglo XX, derivados del efecto del año 2000 y del euro favorecieron el desarrollo de nuevos productos, llamados ERP. Realmente no existe ninguna diferencia, en la práctica, entre los ERP y los antiguos sistemas MRP II y su nombre responde más a una estrategia comercial

1.1 Objetivos

El principal objetivo del proyecto es el despliegue e implantación de un sistema integrado de gestión (ERP) dentro de la empresa. Los ERP son soluciones informáticas cuyo objetivo es gestionar la información a través de las diferentes áreas de la empresa para agilizar tareas, mejorar los procesos de producción y reducir costes.

La aplicación que se utilizará en el proyecto es Microsoft Dynamics NAV uno de los sistemas que en la actualidad se está implantando en muchas empresas dentro del marco de las medianas empresas.

Navision se adapta a los requerimientos del mercado tanto en el momento de la instalación como a lo largo del tiempo cuando las necesidades de negocio se deben variar inevitablemente. El propio desarrollador puede predeterminar los criterios de clasificación y selección de datos que se presentan en pantalla.

La integración entre los diferentes módulos se realiza en tiempo real y de forma automática. Navision incorpora una serie de módulos estándar con los que los desarrolladores construyen la aplicación a la medida de las necesidades de la empresa.

Con Navision no se perderá ningún dato si se produce fallo eléctrico, gracias a su sistema de seguridad que también protegerá la información frente a sustracciones no deseadas con la definición de usuarios y la utilización de contraseñas.

Todas estas características unidas a la capacidad de integración con otras aplicaciones hacen de Navisión un sistema muy práctico y útil para mantener la gestión de la empresa siempre al día y para extraer la información necesaria a la hora de tomar decisiones importantes sobre el desarrollo de la misma.

Con esta herramienta se pueden diseñar procesos automatizados para mejorar la gestión del almacén, contabilidad, gestión de clientes y proveedores y que el flujo de los datos sea rápido y eficaz provocando que se multipliquen los beneficios de la empresa.

1.2 Solución Desarrollada

Para esta solución hay que tener en cuenta dos vertientes: Por un lado se comentará la solución que se ha propuesto para poder tener los datos de las tiendas en tiempo real. Por el otro lado, se comentará la forma de trabajar que se tiene en el almacén central.

En lo referente a tiendas, en primer lugar se mostrará un diagrama dónde se muestra cómo están comunicadas las tiendas con el almacén central. A partir de ese diagrama se explicará cómo se envían los datos desde la tienda a central.

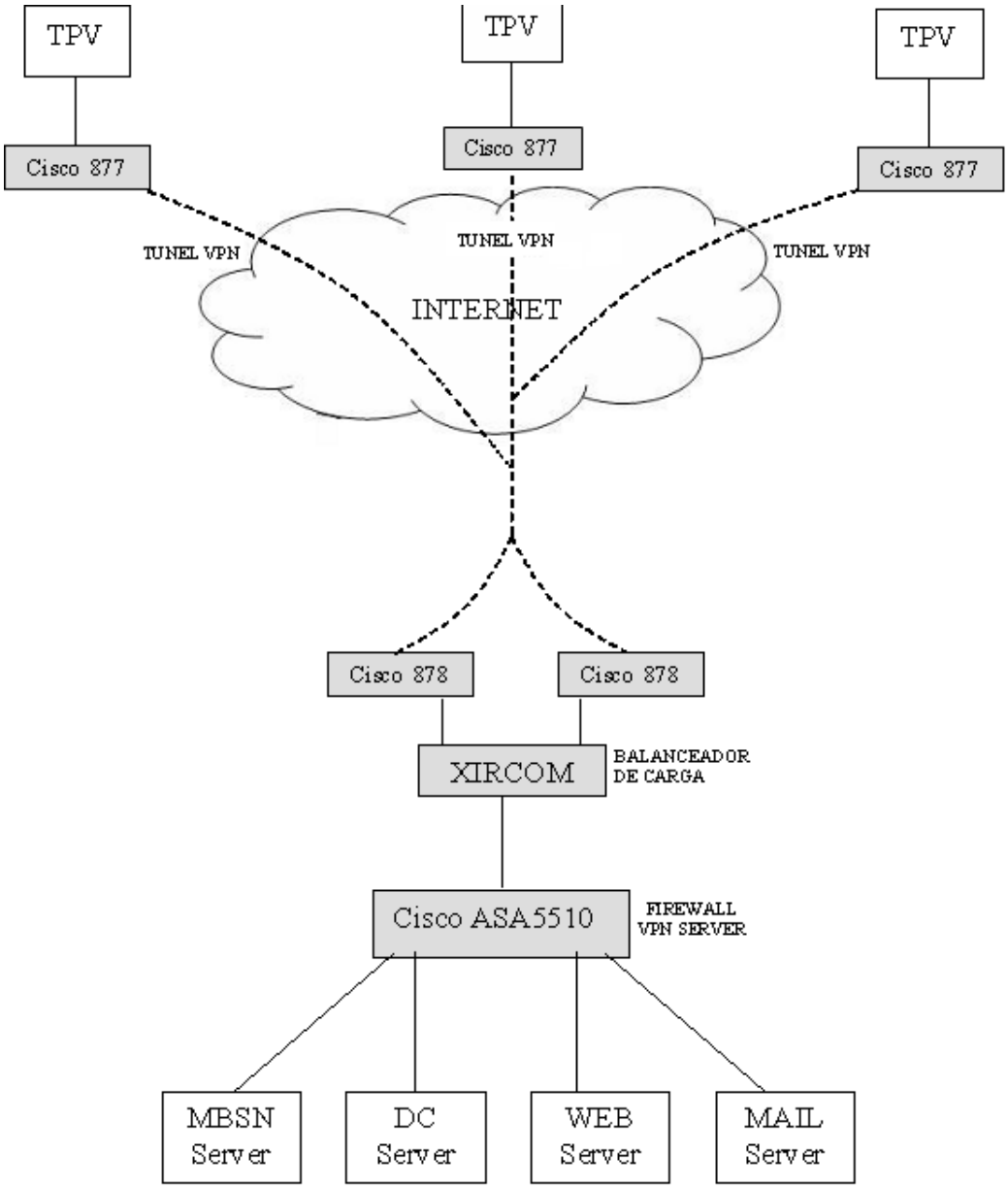


Ilustración 1 Conexión en Tienda

En las tiendas, se tiene instalado en su ordenador el TPV para poder realizar las diferentes operaciones que se le permiten hacer según el usuario que lo esté utilizando en ese momento, ya que lo puede estar utilizando un vendedor o el encargado de la tienda. El vendedor puede realizar las siguientes funciones: realizar una venta, realizar una reserva de un producto al cliente, crear un nuevo cliente y obtener información del producto a través de la ficha.

El encargado puede realizar las siguientes funciones: todo lo mencionado anteriormente en el perfil de vendedor y, además, es el encargado de realizar devoluciones, de hacer pedidos manuales para que almacén central se lo prepare, puede

realizar transferencias de mercancías entre tiendas y es el encargado de realizar el cotejo de la mercancía que le ha llegado a la tienda.

Todas las operaciones descritas se deben actualizar en tiempo real en almacén central, para que almacén tenga los pedidos informatizados y así poder gestionarlos, el departamento de compras pueda saber lo que se vende en cada tienda y, en función de esas ventas, pueda repartir la mercancía y el departamento de contabilidad pueda saber también las ventas que se están produciendo en cada tienda y ver los beneficios que está generando.

Las operaciones se deben tratar cómo paquetes o mensajes para que se puedan gestionar de forma correcta por la red y puedan llegar sin problemas al servidor y se registre en el sistema.

A continuación, se explica cómo viajan esos paquetes por la red para que lleguen de forma satisfactoria al servidor:

Las tiendas están provistas de router Cisco 877 los cuales enlazan con la central a través de un túnel VPN cifrado, con el cual se controla además del tráfico de entrada a la central, el uso de la red que hacen las tiendas.

La central dispone de líneas SHDSL con router Cisco 878 y dos ADSL con router Cisco 877, conectadas entre sí a un balanceador de carga Xircom, el cual asegura la comunicación continua ante cualquier eventualidad. Este balanceador se une a un Firewall/VPN Cisco ASA5510, que es el encargado de asegurar las comunicaciones estableciendo el cifrado de las mismas, así como de gestionar el tráfico de la red y los distintos usos que se dará a la misma, Web, Mail, VPN, etc.

En lo referente al almacén central, se conecta cada uno de los puestos con el servidor de Navision. A continuación se explicará como funciona la red dentro de la empresa.

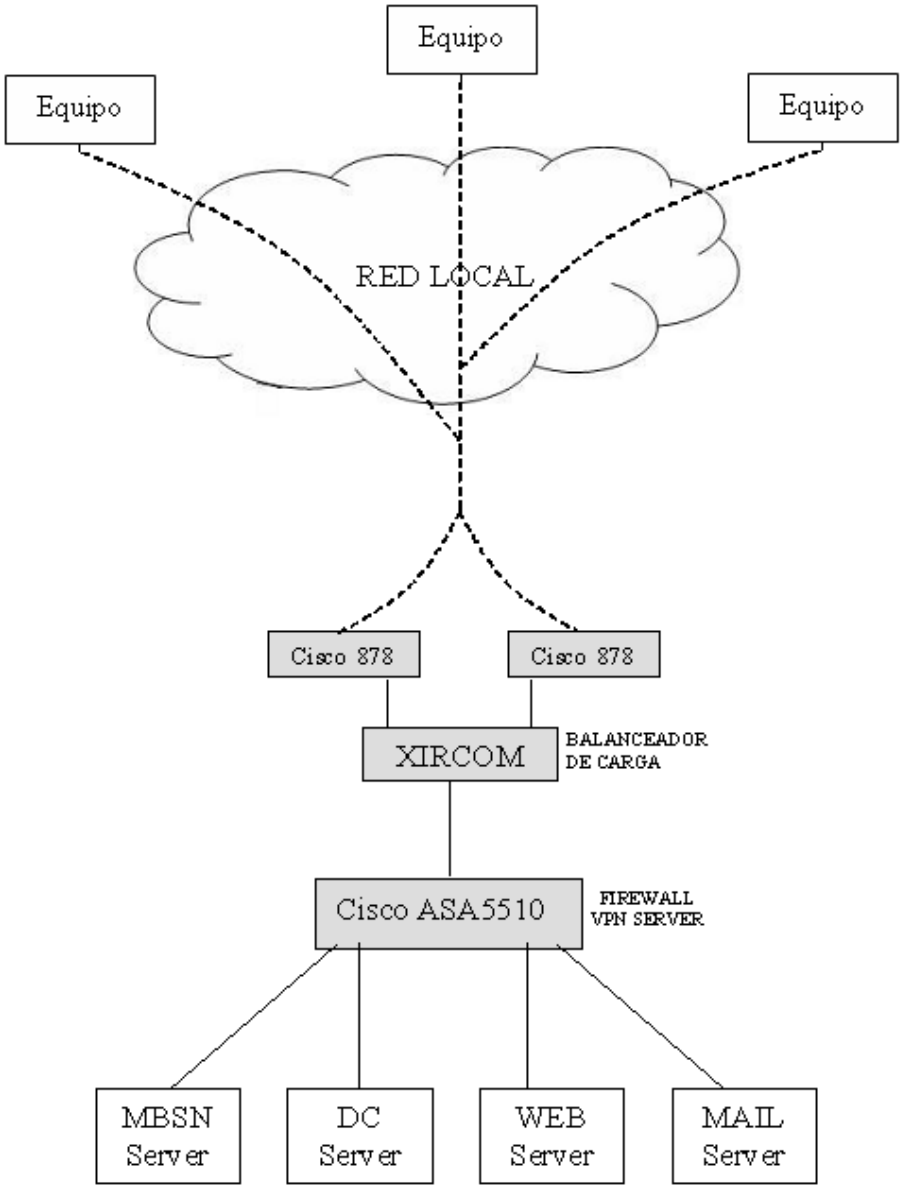


Ilustración 2 Conexión en Central

Cómo se observa en la ilustración 2, al estar tanto Navision como los diferentes puestos de almacén en la misma red local, la forma de conexión con el servidor es mediante LAN. Por lo tanto, mediante LAN se enviarán los paquetes que vayan generando cada uno de los puestos de trabajo para que se pueda registrar en la base de datos. Por otro lado, gracias a que se dispone de líneas SHDSL con router Cisco 878 y dos ADSL con router Cisco 877, conectadas entre sí a un balanceador de carga Xircom, el cual asegura la comunicación continua ante cualquier eventualidad. Este balanceador se une a un Firewall/VPN Cisco ASA5510, que es el encargado de asegurar las

comunicaciones estableciendo el cifrado de las mismas, así cómo de gestionar el tráfico de red y los distintos usos que se dará a la misma, Web, Mail, etc.

A continuación, se exponen las diferentes funcionalidades que se realizan en cada uno de los departamentos existentes en la empresa:

Compras: Las principales funciones de este departamento son:

Pedidos de Compra: Se deben crear nuevos pedidos de compra dentro de Navision. Entonces, una vez que es validado y registrado, se crea un albarán de dicho pedido. En determinadas ocasiones, esos pedidos de compra van directamente a tienda y se registra directamente el pedido para cargar el stock en la tienda que corresponda.

Albarán de Compra: Una vez recepcionado el pedido de compra y con el albarán en la mano, se va a dicho pedido y se rellena el campo cantidad recibida de cada una de las referencias que hayan traído en esa recepción.

Reparto: Una vez hablado con el jefe de compras de cómo se quiere hacer el reparto de esa mercancía, ya que existen diferentes maneras: en función de las ventas, por stock mínimo o reparto manual. Se va albarán correspondiente, se ejecuta el reparto y se imprime para dárselo a los responsables de almacén para que coloquen el reparto.

Incidencias: En este caso, se refiere a cuando llega un pedido de compra y ha venido con menos unidades de las pedidas. En ese caso, se realiza un documento de incidencia para enviárselo al proveedor y que tenga constancia de lo ocurrido. La incidencia se realiza de la misma manera que un pedido de compra, lo único que cambia es que se pone en negativo las unidades.

Contabilidad: Las principales funciones de este departamento son:

Pago: En lo referente a pedidos, una vez que se factura un pedido, se crea una factura con la fecha de vencimiento correspondiente a ese proveedor. Entonces, cuando se vaya a realizar el pago de esa factura, se debe dejar constancia de ese hecho en Navision. Para realizar eso, se debe hacer mediante el diario de pagos donde se indica qué número de factura se quiere liquidar.

Por otro lado, otra función del diario de pagos es la de liquidar las compras hechas por el departamento de informática, recibos de la luz, recibos de teléfono, etc.

Facturas: En este caso, se encargan de crear una nueva factura donde se indique al proveedor que se le debe hacer el pago, la forma de pago, la fecha de vencimiento de dicha factura y los productos. Cuando llega el momento de pagarla se debe liquidar esa factura mediante el diario de pagos.

Abono: Existen dos tipos: abonos de proveedores y abonos de clientes. Los abonos de proveedores pueden deberse a varias circusntancias: devolución de productos en mal estado, acuerdos de descuento en los productos o descuentos por pronto pago. Para ello, se crea un nuevo abono donde se indique a qué proveedor corresponde, la forma de pago, fecha de vencimiento, productos y un comentario indicando el motivo del abono. Para el caso del cliente, ocurre como en el caso del proveedor, que existen varios motivos: devolución del importe de un pedido Web, devolución del importe de un pedido hecho por una franquicia, devolución del importe hecho a un cliente de mayor. Para ello, se creará un nuevo abono donde se indique a qué cliente corresponde, la forma de pago, fecha de vencimiento, productos y un comentario indicando el motivo del abono.

Presupuestos: Son los encargados de definir cual es el presupuesto, fraccionado en meses, que se tiene para realizar la compra de los diferentes productos. En cada mes, está desglosado el presupuesto que hay destinado para cada familia de producto. Por lo tanto, cuando se realiza un pedido de compra de esa familia y se valida el pedido, el programa comprueba si hay suficiente dinero para realizar esa compra teniendo en cuenta la fecha de vencimiento de dicho pedido. Si hay suficiente se valida sin problemas. En caso de que no haya suficiente, se debe negociar la fecha de vencimiento con el proveedor para hacer frente al pago.

Almacén

Pedidos: Se debe tener en cuenta que existen varios tipos de pedidos. Los pedidos que realizan las tiendas, los pedidos que se crean al procesar las ventas

y los pedidos que crea manualmente el personal de almacén para introducir productos en un pedido de implantación. A continuación, se explica brevemente la funcionalidad de cada uno de los pedidos. En lo referente a pedidos de tienda, el TPV tiene una funcionalidad donde el encargado de la tienda realiza pedidos manuales al almacén. Una vez que llega el pedido a central, se analiza cada línea del pedido y se comprueba a que familia de productos corresponde. Una vez comprobado a que familia pertenece, se comprueba si hay algún pedido de esa familia abierto. Si existe un pedido, se inserta esa línea al pedido. En caso de no existir, se crea un nuevo pedido de esa familia. Este tipo de pedidos son pedidos manuales.

En lo referente a los pedidos que se crean mediante el procesamiento de ventas, tienen la siguiente funcionalidad: se procesan las ventas que se han tenido en cada una de las tiendas, se comprueban los productos que se han vendido en ese día y se sigue la siguiente fórmula: en la tienda debe tener el suficiente stock como para vender lo que se ha vendido el día anterior durante los próximos tres días. En caso de que no haya suficiente stock se crea una línea de pedido. Una vez creada la línea se comprueba si hay algún pedido abierto correspondiente a la familia del producto. Si existe, inserta esa línea en el pedido. En caso de no existir se crea un nuevo pedido de esa familia. Este proceso se realiza hasta que se hayan procesado las ventas de todas las tiendas. Este tipo de pedidos son pedidos automáticos.

En lo referente a pedidos que abren los responsables de almacén, se realiza cuando se abre una nueva tienda y se debe hacer un pedido de implantación a la tienda. Este tipo de pedidos son pedidos manuales.

Registro Pedidos: En este caso, se explica cómo se registran los diferentes pedidos que existen: para el caso de pedidos manuales de tienda se prepara y se registra de la siguiente manera: El usuario entra en el pedido, mediante una PDA, y el sistema le marca a que ubicaciones debe ir para coger el producto que indica en la pantalla. En caso de que el producto se encuentre en esa ubicación, el operario debe coger las unidades que marca e indicar en la PDA cuántas ha cogido. Una vez marcado eso, la PDA manda un mensaje al sistema para que registre ese movimiento. Esta operación se debe realizar hasta que la PDA marque que no tiene ninguna orden pendiente. Una vez terminado de

poner el pedido, debe ir al ordenador a registrar el albarán que ha ido creando para cargar esos productos en el stock de la tienda. Por otro lado, se imprime la transferencia para que vaya junto al palet con la mercancía para que sepan que esos productos pertenecen a ese pedido. Para el caso de pedidos automáticos que se han generado de procesar las ventas o pedidos de implantación, el proceso es análogo al que se realiza en los pedidos manuales de tienda.

Etiquetas: En este apartado se tienen dos opciones: Sacar las etiquetas de un albarán que se haya ejecutado el reparto e ir poniendo las etiquetas a los productos y depositar en los boxes de las tiendas los productos que se les ha asignado a cada una de ellas.

1.3 Planificación

La metodología seguida en el desarrollo del proyecto podría considerarse básica, habiendos realizado las siguientes fases:

Recopliar Información: En este apartado se realiza la búsqueda de toda la información existente acerca de los ERP. En primer lugar, se busca información acerca de toda la historia referente a los ERP, es decir, su nacimiento y posterior evolución a lo largo de los años. A continuación, se busca información de los principales ERP existentes en el panorama PYMES, para poder explicar cada uno de ellos de manera detallada.

1. Introducción: En este apartado se explica, de mánera genérica, en que va a consistir el proyecto final de carrera, la solución que se va a desarrollar y enumerar los diferentes apartados del proyecto.
2. Estado de los ERP: En este apartado se explica, de manera detallada, los ERP más utilizados en la pequeña y mediana empresa. A la hora de explicar cada uno de ellos, se enumera sus múltiples características y las diferentes funciones que se realizan.
3. Realizar Aplicación del Proyecto: Este apartado corresponde a la parte práctica del proyecto, es decir, el desarrollo del módulo de ventas para las tiendas. En primer lugar, se debe recopilar información referente al lenguaje de programación asociado a Navision, para poder saber los principios básicos del lenguaje y así empezar a desarrollar la aplicación. Una vez sabidos los conceptos, se pasa a desarrollar y probar cada una de las funciones que tendrá la aplicación, en este caso son: Caja, proceso de ventas, almacen e informes.

4. Memoria del Trabajo Realizado: En este apartado se explica, de manera detallada, cada uno de los módulo existentes en el programa. En cada uno de ellos se habla de las principales funciones que se pueden realizar y los movimientos que se producen en cada una de ellas.
5. Estructura del Trabajo Realizado: En esta apartado se explica el trasiego de información que se produce entre los diferentes módulos con el programa. En cada módulo, se muestra la información entrante y el resultado generado. Por otro lado, por cada módulo se hace una descripción de alto nivel, es decir, para las principales funciones del módulo se muestra el código y el pseudocódigo.
6. Seguridad y Base de Datos: En este apartado se explica todo lo que corciene a la seguridad de Navision, es decir, las restricciones que se pueden aplicar a nivel usuario, a nivel de contraseñas, a nivel de roles o a nivel de registros. Además, en lo referente a la base de datos se explica todos los pasos seguidos para configurar la base de datos de la aplicación. En este caso, los pasos seguidos son: realizar diagrama de sistema, realizar diagrama de flujo de datos, explicar los diferentes flujos de datos, explicar las tablas utilizadas, mostrar y explicar las relaciones entre tablas y hablar sobre el motor utilizado en la base de datos.
7. Temas Finales: En este apartado se explican una serie de cosas, que son: Conclusiones sacadas gracias a la realización del proyecto, futuras líneas para mejorar la aplicación realizada, mostrar la bibliografía utilizada en la realización del proyecto y el presupuesto asociado a la realización del proyecto.

Id		Nombre de tarea	Duración
1		Inicio del Proyecto	0 días
2		**Recopilar Información**	**28 días**
3		Historia de los ERP	7 días
4		Información SAP	7 días
5		Información Axapta	7 días
6		Información Navision	7 días
7		**Introducción**	**11 días**
8		Redactar Introduccion General	2 días
9		Redactar Objetivos	1 día
10		Busqueda Informacion Comunicaciones	2 días
11		Redactar Solución Comunicaciones	2 días
12		Redactar Caracteristicas Módulos	4 días
13		Redactar Conclusiones	2 días
14		Analizar y Especificar Fases del Proyecto	2 días
15		Realizar Diagrama de Gantt	2 días
16		Redactar Estructura del Documento	1 día
17		**Estado de los ERP**	**13 días**
18		**Sistemas ERP**	**4 días**
19		Redactar Características Sistemas ERP	4 días
20		**Sistema SAP**	**3 días**
21		Redactar Características Sistema SAP	3 días
22		**Sistema Axapta**	**2 días**
23		Redactar Caracteristicas Sistema Axapta	2 días
24		**Sistema Navision**	**3 días**
25		Redactar Características Sistema Navisic	3 días
26		Descripcion Arquitectura del Sistema Navisior	3 días
27		**Realizar Aplicación del Proyecto**	**360 días**
28		Estudiar Lenguaje de Programación	21 días
29		Configurar Base de Datos	4 días
30		Creación de Tablas	2 días
31		Desarrollar Módulo Caja	14 días
32		Realizar Pruebas Módulo Caja	4 días

Id		Nombre de tarea	Duración
33		Desarrollar Módulo Proceso Ventas	60 días
34		Realizar Pruebas Módulo Proceso Ventas	7 días
35		Desarrollar Módulo de Informes	21 días
36		Realizar Pruebas Módulo de Informes	5 días
37		Desarrollar Módulo de Almacén	70 días
38		Realizar Pruebas Módulo de Almacén	7 días
39		Desarrollar Módulo de Mantenimiento	7 días
40		Realizar Pruebas Módulo de Mantenimiento	1 día
41		Desarrollar Funciones Servicio Web	90 días
42		Realizar Pruebas Funciones Servicio Web	10 días
43		Configurar Menú Informática	30 días
44		Realizar Pruebas Menú Informática	6 días
45		Configurar Usuarios	3 días
46		**Memoria del Trabajo Realizado**	**17 días**
47		Describir módulo Almacen	5 días
48		Describir módulo de Compras y Pagos	2 días
49		Describir módulo de Ventas y Cobros	2 días
50		Describir módulo de Contabilidad	4 días
51		Describir módulo de Gestión de Tiendas	3 días
52		Describir Servicio Web	3 días
53		**Estructura del Trabajo Realizado**	**54 días**
54		**Arquitectura de la Aplicación**	**3 días**
55		Realizar Arquitectura del Sistema	2 días
56		Describir Flujo del modulo Contabilidad	1 día
57		Describir Flujo del modulo Ventas y Cobro	1 día
58		Describir Flujo del modulo Compras y Pag	1 día
59		Describir Flujo del modulo Almacen	1 día
60		Describir Flujo del modulo de Tiendas	1 día
61		**Descripción Alto Nivel de cada módulo**	**44 días**
62		**Almacen**	**9 días**
63		Describir Funcionalidad	5 días
64		Describir Algoritmos	4 días

Id		Nombre de tarea	Duración
65		**Compras y Pagos**	**7 días**
66		Describir Funcionalidad	4 días
67		Describir Algoritmos	3 días
68		**Ventas y Cobros**	**7 días**
69		Describir Funcionalidad	4 días
70		Describir Algoritmos	3 días
71		**Contabilidad**	**5 días**
72		Describir Funcionalidad	3 días
73		Describir Algoritmos	2 días
74		**Tienda**	**13 días**
75		Describir Funcionalidad	7 días
76		Describir Algoritmo	6 días
77		**Comunicaciones**	**3 días**
78		Describir Funcionalidad	2 días
79		Describir Algoritmos	1 día
80		**Manual de Usuario**	**7 días**
81		Definir Apartados	2 días
82		Realizar Manual	5 días
83		**Seguridad y Base de Datos**	**9 días**
84		Definir Seguridad de Navision	2 días
85		Definir Base de Datos de Navision	7 días
86		**Temas Finales**	**3 días**
87		Redactar Conclusiones	1 día
88		Redactar posibles mejoras	1 día
89		Indicar Bibliografía	1 día
90		Realizar Presupuesto	1 día
91		Fin del Proyecto	0 días

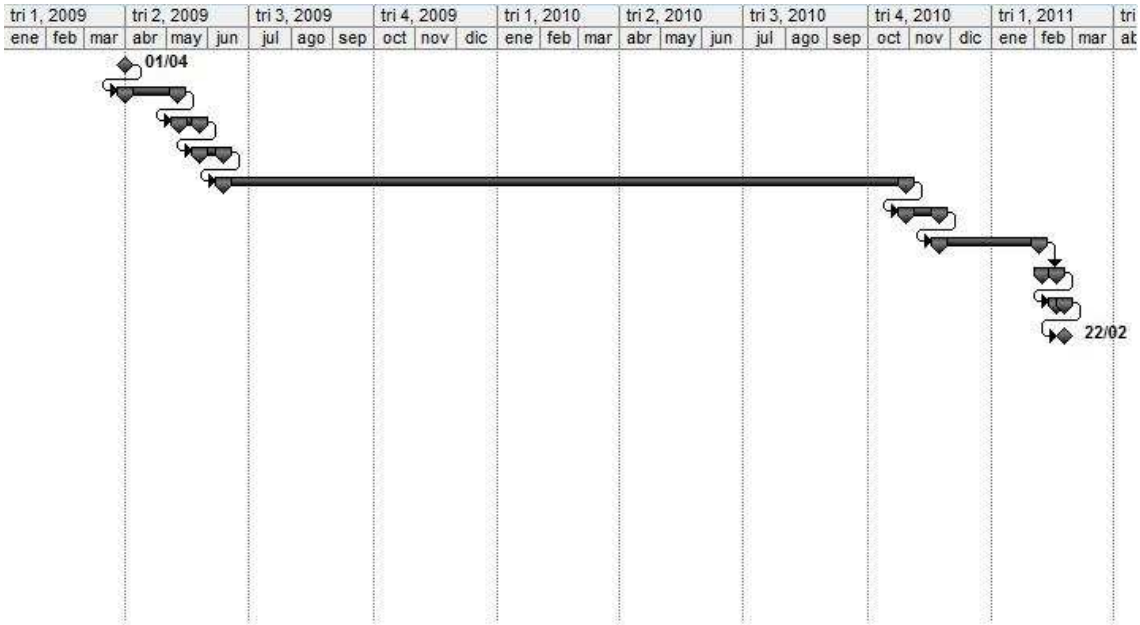


Ilustración 3 Diagrama de Gantt

1.4 Estructura del documento

Capitulo 2, Estado de los ERP. En este apartado se hablará de cómo están en la actualidad los diferentes programas de sistemas integrados de gestión, es decir, se hablará de forma breve acerca de Navision, SAP, Axapta, etc. Se explicarán las características más importantes que tiene cada uno de los programas que tratan sobre el ERP.

Capítulo 3, Planteamiento. En este capítulo se detallan el diseño y la construcción de la interfaz. Se describe la arquitectura de la aplicación con especial atención a las funcionalidades añadidas y los manuales de usuario y de referencia.

Capítulo 4, Estructura del Trabajo. En este apartado se relata cómo está estructurado cada uno de los módulos del sistema, donde se explicarán en cada uno de ellos las principales funciones y se hará mediante pseudocódigo.

Capítulo 5, Seguridad y Base de datos. En este apartado se relata todo lo que concierne a la seguridad del programa, es decir, se hablará de cómo asignar roles a los usuarios para que accedan a una serie de pantallas, de como asignarles a qué menús deben acceder y que en cada registro queda constancia del autor del mismo. En lo referente a la base de datos, se hablará de su estructura y configuración.

Capítulo 6, Conclusiones. En este capítulo se detallan las conclusiones principales sobre el software así como algunas consideraciones personales acerca del entorno de fuente abierta en el que se ha desarrollado el Proyecto.

Capítulo 7, Futuras líneas de trabajo. En este capítulo, se describen con cierto detalle las posibles líneas futuras de trabajo de extensión de las funcionalidades tanto del programa como de los módulos que se han desarrollado.

2 Estado de los ERP

En este apartado, se explica en primer lugar las características generales de los sistemas ERP. A continuación, se habla de forma breve sobre los sistemas ERP más importantes que se utilizan, cómo pueden ser SAP. Además, se habla de AXAPTA que aún no se le puede catalogar de ERP importante, pero poco a poco ha ido teniendo importancia dentro del panorama nacional. Por último, se expone de forma más detallada: las características, el lenguaje de programación que se utiliza y las principales utilidades que tiene el programa Microsoft Business- Solutions NAV.

2.1 Sistemas ERP

Los sistemas de planificación de la empresa son sistemas de gestión de información que integran y automatizan muchas de las prácticas de negocios asociadas con los aspectos operativos o productivos de la empresa

Estos sistemas se caracterizan por estar compuestos por diferentes partes integradas en una única aplicación. Estas partes pueden ser: compras, ventas, contabilidad, inventarios, control de almacén, recursos humanos, etc. El ERP integra todo lo necesario para el funcionamiento de los procesos de negocio de la empresa.

En lo referente a la configuración del ERP es compleja, debido a que no existe una configuración estándar del ERP, sino que es particular para cada empresa. Por lo tanto, a la hora de configurar hay que tener en cuenta los siguientes factores:

- **Definir los resultados a obtener con la implantación del ERP**: En este caso se debe definir lo que queremos obtener al implantar el ERP en la empresa, ya que a partir de los resultados definidos, se puede ir configurando el ERP

- **Definición del modelo de negocio**: Se debe explicar cómo queremos enfocar el negocio, es decir, se deben definir cuales deben ser los principales activos de la empresa.

- **Definición del modelo de gestión**: Se deben definir las características de la empresa y, además decidir de que manera se quiere gestionar la empresa, es decir, si se quiere tratar como una pequeña empresa, mediana empresa o una empresa grande. A partir de esa decisión, se deben definir las pautas de gestión del almacén, así como los principales procesos a realizar dentro del mismo.

- **Definición de la estrategia de implantación:** En este apartado, se debe definir cómo está estructurada la empresa. Una vez definido ese aspecto, se debe estudiar el modelo de implantación que se quiere hacer en la empresa. Lo recomendable en estos casos es instalar un servidor, donde se aloja la base de datos. En cada uno de los puestos de la empresa se instala el cliente del programa, el cual se conectará al servidor para tener en tiempo real los datos actualizados.

- **Auditoría del entorno técnico y del entorno de desarrollo:** Se debe analizar cómo está estructurada la empresa en cuanto a comunicaciones y equipos existentes en la misma. Se debe realizar un estudio sobre si se debe invertir en mejorar esos componentes para que la implantación resulte satisfactoria o por el contrario, no se necesita una inversión fuerte debido a que con el material existente se puede realizar una buena implantación

- **Controles de Calidad:** Antes de realizar la implantación del sistema, se debe comprobar si todo el desarrollo realizado satisface las necesidades de la empresa y además, ayuda a agilizar el flujo de almacén, lo que produce que aumente los beneficios de la empresa. En caso de que los controles de calidad no sean del todo satisfactorios, se debe de reprogramar el desarrollo para adecuarse a las necesidades empresariales.

Por último, se explica las ventajas que se tiene al implantar un sistema ERP:

Una empresa que no tenga implantado un ERP tiene el problema de que existen una serie de aplicaciones software que no se pueden personalizar ni optimizar para su empresa. Por lo tanto, no se puede hacer un diseño de ingeniería para la mejora del

producto, seguimiento del cliente desde la aceptación hasta la satisfacción completa. Con los sistemas ERP se dispone de estas utilidades

Por otro lado, la seguridad a los accesos está incluida en el ERP para poder proteger a la empresa de delitos externos, tales como el espionaje industrial, delito interno o malversación

Hay conceptos de mercadeo y ventas que incluyen control de calidad, para asegurar que no hay problemas no solucionados en los productos finales.

2.2 SAP

SAP ayuda a gestionar y planificar la utilización de los recursos mediante sistemas de gestión ERP. Estos sistemas permiten a los clientes tener un completo control de la utilización de todos los recursos disponibles en la empresa. Además, permite a los clientes cuantificar sus cantidades y planear respectivamente la expansión de sus bases utilizando estimados fiables.

SAP pone a disposición de las empresas numerosas funcionalidades que les permiten analizar su negocio, optimizar sus finanzas, gestionar sus recursos humanos, operaciones y servicios corporativos. Además, les proporciona soporte para cuestiones referentes a la gestión de sistemas de administración de usuarios, gestión de configuración, gestión de datos centralizada y gestión de servicios WEB. Todas estas funcionalidades se integran dentro de la plataforma SAP NetWeaver, lo que permite a los usuarios medir y gestionar sus recursos empresariales en tiempo real.

Las principales características de esta aplicación son:

- **Horizontalidad / Verticalidad**: Esta aplicación está orientada a todo tipo de actividades y sectores en general. Por lo tanto, existe un grado muy alto de horizontalidad.

- **Escalabilidad:** La aplicación está implementada sobre una plataforma escalable, es decir, se permite agregar nuevos campos de datos sobre la marcha sin necesidad de ayuda técnica.

- **Modularidad:** La aplicación permite estructurar distintos módulos relacionados entre sí, aunque en el momento de la implantación se realiza como un único núcleo. Además, una vez implantado, no permite agregar nuevos módulos.

- **Interconexión entre módulos:** Esta aplicación dispone de una alta interconexión entre módulos, es decir, permite la relación de información entre un módulo y otro, obteniendo un análisis instantáneo de facturas, ofertas u otros datos relacionados

- **Parametrización:** La aplicación permite una personalización alta en el producto, es decir, se le pueden agregar sus propios campos o tablas, a los que se les pueden enlazar diferentes casillas desplegables y procesos

- **Accesibilidad:** SAP está desarrollado para el entorno informático WINDOWS, soportando sistemas operativos WINDOWS 2000, WINDOWS XP y WINDOWS VISTA Y WINDOWS 7.

- **Facilidad de Uso:** La aplicación tiene un entorno de WINDOWS intuitivo y dispone de un asistente que guía al usuario paso a paso a través de la definición de los parámetros necesarios para una consulta. Además, permite un acceso fácil a la base de datos.

2.2.1 Características Funcionales

En este apartado se describe la situación actual de la herramienta evaluada desde el punto de vista de las características funcionales y de la profundidad con la que se tratan, orientadas a gestionar las actividades más comunes de la empresa, cómo son ventas, contabilidad, clientes y proveedores.

Contabilidad

En este apartado, se trata de informatizar cada uno de los movimientos contables que se realizan en la empresa, es decir, cuando se realiza un pedido de compra y se ha dado por entregado y facturado, el sistema crea una factura que está pendiente de pagar al proveedor. Entonces, en ese instante se crea un movimiento contable de compra. Todo ese proceso, gracias a SAP, queda informatizado.

Por otro lado, para el tema de ventas a clientes o a franquiciados ocurre exactamente lo mismo que en las compras, es decir, se realiza un pedido de venta al cliente que le corresponda. Una vez que el cliente nos comunica que le ha llegado el pedido, se debe dar como recibido y facturado el pedido. Una vez que se ha facturado, se crea una factura pendiente de abonar por el cliente. Entonces, en ese instante se crea un movimiento contable de venta. Todo ese proceso, gracias a SAP, queda informatizado.

Además, cuando llega el momento de realizar el pago al proveedor, ese proceso debe quedar informatizado. Mediante el diario de pagos, se busca el documento que se debe pagar al proveedor, una vez buscado se realiza el asiento contable y se registra ese movimiento. En ese momento, se crea un movimiento contable de pago.

Por otro lado, cuando llega el momento de recibir el pago de un cliente, ese proceso debe quedar informatizado. Mediante el diario de cobros se busca el documento que se debe cobrar del cliente, una vez buscado se realiza el asiento contable y se registra ese movimiento. En ese momento, se crea un movimiento contable de cobro.

Además, la empresa puede definir su propio plan de cuentas, es decir, el programa lleva un plan de cuentas estándar pero ese plan de cuentas se puede moldear al gusto de la empresa, ya que cada empresa gestiona sus cuentas de una manera particular. Además, cuando se registran movimiento de pagos, cobros, compras, ventas, etc. la empresa puede decidir a que cuentas se quieren cargar cada uno de esos movimientos para luego entrar en el banco correspondiente a esa cuenta y cotejar que todo lo que recibe el banco es correcto.

Ventas

En este apartado, se explica de forma breve cómo es el proceso informático para el tema de las ventas de la empresa. Existen dos formas de hacerlo: si para la venta que ha

pedido el cliente existe stock en el almacén, se crea una factura donde se indica el cliente al que se factura, el vendedor que lo realiza, la forma de pago de la factura. Una vez rellenado todos los datos del cliente, se debe empezar a insertar los productos que ha pedido el cliente. En caso de ser un cliente especial, aparecerá el descuento que se realiza en cada uno de los productos. Una vez se hayan puesto todos los productos, se debe registrar esa factura para que se produzca el correspondiente movimiento contable y además, quede plasmado que es una factura pendiente de cobrar. A la hora del cobro de la factura, mediante el diario de cobros se crea el movimiento contable del cobro de dicha factura y se registra, para que quede informatizado que se ha cobrado el importe de la factura.

La otra forma es la siguiente: En caso de que en el pedido que ha realizado el cliente existan una serie de productos de los que en ese momento no hay stock en el almacén, se crea un pedido con los productos que nos ha indicado. Una vez creado el pedido, se registra para que quede constancia de que se tiene que servir al cliente. En el momento en que tenemos existencias de esos productos, se debe servir el pedido. Una vez servido al cliente, se debe crear la factura de cobro correspondiente a ese pedido. En este caso, solo se rellena la ficha con los datos del cliente, ya que existe una opción donde se indica el pedido del que se quieren volcar las líneas, para que en la factura aparezca a que pedido corresponde. Una vez se haya terminado de hacer la factura se debe registrar la misma, para que se produzca el correspondiente movimiento contable y además, queda plasmado que es una factura pendiente de cobrar. A la hora del cobro de la factura, mediante el diario de cobros, se crea el movimiento contable del cobro de dicha factura y se registra, para que quede informatizado que se ha cobrado el importe de la misma.

Compras

En este apartado, se explica, de forma breve, cómo es el proceso informático para el tema de las compras de la empresa. En primer lugar, se debe crear un pedido de compra, donde se indican los datos del proveedor al cual se quiere realizar el pedido. Una vez se han rellenado los datos pertinentes, se debe empezar a indicar los productos que se quieren pedir y las cantidades correspondientes. Una vez acabado con ese proceso, se debe validar el pedido para ver si hay el suficiente presupuesto para hacer frente al pago del pedido. Si hay suficiente presupuesto, se valida sin problemas el

pedido. En caso contrario, emite un mensaje diciendo que en ese mes no existe suficiente presupuesto. En ese caso, se debe negociar con el proveedor la fecha del pago. A continuación, cuando llega la mercancía de ese pedido al almacén se debe indicar, mediante el albarán de compra, las unidades que han llegado de cada producto. Una vez hecha esa operación, se debe registrar el pedido para que se cree la factura correspondiente al pago de ese pedido al proveedor. Se debe tener en cuenta que si no ha llegado toda la mercancía que se pidió, se debe crear un pedido de incidencia indicando para cada producto la diferencia de unidades entre las cantidades pedidas y recibidas. Una vez terminado ese proceso, se debe registrar para que se cree el abono correspondiente, indicando el pago que hace el proveedor a la empresa por esa incidencia en la entrega.

Almacén

En este caso, se explican las principales funciones que deben realizar las personas responsables de almacén. Son los encargados de recepcionar la mercancía que se ha ido pidiendo a los proveedores. Una vez que se haya recepcionado toda la mercancía, deberán entregar el albarán para que puedan registrar el pedido y una vez registrado, el sistema le indique dónde debe ubicar cada uno de los productos, es decir, el sistema crea dos acciones, en una indica la cantidad y la ubicación de ese producto para reserva y la otra acción indica la cantidad y la ubicación de ese producto para la preparación de pedidos. Por otro lado, también son los encargados de preparar los pedidos. Entonces, cuando el pedido puede prepararse al completo, el programa emite una serie de acciones donde indica en cada una de ellas: producto, cantidad que debe preparar y dónde se encuentra el producto. Una vez que se haya terminado de preparar el pedido, se da por cerrado el pedido y se imprime el packing list del mismo para que el destinatario compruebe que le ha llegado todo.

[SAP] SAP. Disponible [Internet]:
http://www12.sap.com/spain/solutions/business-suite/features-functions/index.epx

2.3 AXAPTA

Axapta es una solución de gestión empresarial para organizaciones de tamaño medio y pequeñas organizaciones que trabajan con el software de Microsoft, para

mejorar la productividad del personal. Axapta facilita la realización de negocios en diferentes ubicaciones y países, consolidando y estandarizando los procesos.

2.3.1 Características

Axapta es un sistema de gestión empresarial integrado y basado en Web, que ofrece la siguiente propuesta: ser verdaderamente dueño de su ERP y sólo pagar por el soporte profesional que necesite. El sistema está enfocado a empresas de pequeño y mediano tamaño. Las características principales del sistema son:

Empleados Eficaces

Axapta permite al personal centrarse en el negocio, en lugar de aprender a usar nuevas aplicaciones, con una solución que se asemeja a la conocida experiencia de usuario de Microsoft. Puede trabajar de forma eficaz gracias a una interfaz de usuario intuitiva y adaptada a roles que ayudan a los empleados a organizar, dar prioridad y obtener acceso a tareas e información desde una única ventana. Axapta permite una fácil integración con las aplicaciones de Microsoft Office, gracias a ello permite al personal trabajar con los datos de Axapta sin salir de las aplicaciones de Microsoft Office.

Crecimiento Exponencial

Axapta permite gestionar correctamente el crecimiento y los cambios mediante la creación de rentables relaciones de la cadena de suministro y la rápida adaptación de procesos internos para satisfacer las demandas cambiantes. Axapta ofrece una arquitectura orientada a servicios, una estrecha integración con las tecnologías de Microsoft y servicios Web que le permiten agregar usuarios de forma sencilla, admitir nuevos modelos de negocio y ampliar una solución para admitir incluso grandes empresas distribuidas. Se puede automatizar sin problemas la colaboración en la cadena de suministro y el intercambio de documentos con socios mediante BizTalk Server y Office System.

Se pueden modificar de manera rápida secuencias de trabajo para aprovechar las nuevas oportunidades y adaptarse con rapidez a los cambios del sector y a las demandas de los clientes. Con las flexibles opciones de licencias empresariales, se puede seleccionar el paquete de soluciones que se necesita y a continuación, agregar capacidades a medida que crece el negocio de forma rápida y rentable.

Competencia

Las oportunidades empresariales pueden significar la ampliación de la base de clientes para incluir destinatarios de todo el mundo o la expansión. Axapta está diseñado para gestionar complejidades de una organización internacional mediante la consolidación y estandarización de procesos. Axapta proporciona expansión del negocio de manera sencilla a cualquier frontera internacional, acoplando el negocio a los métodos financieros de dicha frontera.

Axapta consolida los datos financieros, operativos y de clientes mientras conserva la información local para crear su ventaja estratégica y relaciones personalizadas con los clientes. Axapta unifica los estándares y ayuda a garantizar la calidad de centralización de los procesos. Por otro lado, se puede gestionar y supervisar el rendimiento mundial y local con herramientas de creación de informes y análisis. Además, se puede realizar implementaciones eficaces y coherentes mediante Sure Step, una metodología global estandarizada y que tiene un conjunto de herramientas que simplifican los procesos de implementación y actualización.

Simplificado en Normativa

Las empresas afrontan una ingente cantidad de requisitos normativos que varían mucho entre regiones y países. Axapta ayuda a reducir el riesgo y la responsabilidad asociados al gobierno corporativo, cumplimiento de normativas e iniciativas de clientes.

Axapta asegura la coherencia con procesos estandarizados y repetibles que pueden agregar de modo confiable datos de cumplimiento. Además, convierte los retos en oportunidades admitiendo las iniciativas de los clientes, incluidas las estrategias de cadena de suministro con capacidades que reducen los residuos de fabricación.

Por otro lado, se centraliza y comparte el gobierno corporativo y las actividades de cumplimiento con el centro, una herramienta integrada para gestionar políticas, procesar documentos e informes. Axapta mantiene un estricto control sobre las actividades de cumplimiento con herramientas de inteligencia empresarial que supervisan los cambios internos, para que puedan responder sin problemas a las solicitudes de información

Gracias a este sistema, reduce el coste y la incertidumbre de la protección de datos de cumplimiento de la empresa.

[Axapta] Microsoft Dynamics Axapta 4.0. Disponible [Internet]:
<http://www.dynamicsit.com.co/pdf/modulos/Guia%20Rapida%20de%20MS%20Dynamics%20AX.pdf>

2.4 Microsoft Dynamics Navision

Una aclaración antes de comenzar con el capítulo. En la actualidad Navision es Microsoft Dynamics NAV pero en sus inicios era Navision Financials, por eso en todo el proyecto se dice Navision en vez de NAV.

Navision es una solución integrada para la gestión del negocio en la mediana y grande empresa. Se puede utilizar Navision para sustituir todo lo que se necesite del sistema actual y se puede seleccionar entre diversas áreas de aplicación: gestión financiera, fabricación, distribución, gestión de relaciones con los clientes, gestión de servicios, comercio electrónico y análisis.

Navision ayuda a dirigir el negocio hacia donde el cliente desee y del modo que desee. Por otro lado, se puede responder rápidamente a la demanda de los clientes, a los cambios del mercado y cuidar el negocio del modo más eficaz posible.

Un Partner Certificado local de Microsoft podrá personalizar el software para satisfacer las necesidades específicas de la empresa.

2.4.1 Módulos de Navision

Gestión Financiera

Las soluciones de contabilidad y finanzas de Navision ayudan al cliente a realizar el seguimiento y analizar su información de negocio. La estrecha integración permite gestionar libro mayor, pagos, cobros, inventario, contabilidad analítica, activos fijos y gestión de bancos, así como realizar conciliaciones bancarias. También puede gestionar los procesos financieros entre múltiples divisas, ubicaciones o empresas.

Activo Fijo

El activo fijo son las propiedades, bienes materiales o derechos que en el curso normal de los negocios no están destinados a la venta, sino que representan la inversión

de capital, patrimonio de una dependencia, entidad en las cosas usadas o aprovechadas por ella de modo periódico, permanente o semi-permanente en la producción, en la fabricación de artículos para venta o la prestación de servicios a la propia entidad, a su clientela o al público en general.

Las principales ventajas que se tienen en Navision con respecto a los activos fijos son las siguientes:

Se puede consultar la amortización acumulada mediante cualquiera de los métodos de seguimiento comunes. Gracias a esto, se pueden proyectar los activos de los libros y así poder tomar mejores decisiones sobre el tiempo más rentable de los activos.

Por otro lado, se pueden utilizar los grupos de cuentas para realizar cambios en un grupo grande de activos y se pueden actualizar los campos relacionados con el activo

Además, se puede realizar el seguimiento para poder analizar los costes de mantenimiento y seguro de cada uno de los activos fijos.

Gracias a estas ventajas, se pueden realizar las siguientes operaciones:

Definir y realizar el seguimiento de los activos por su descripción, ubicación o número. Agregar campos personales y establecer vistas predeterminadas, como se muestra en la ilustración 4, donde se observa un ejemplo de definición de activos fijos.

Determinar el nivel de detalle que se necesite en los informes de activos fijos y definir si los activos aparecen como una entidad única o como un conjunto de componentes.

Implementar un método de amortización personalizado que satisfaga los requisitos legales o contables de la empresa.

Registrar el servicio y mantenimiento rutinarios realizados en los activos para supervisar el coste del seguro. Utilizar esos datos para tomar decisiones importantes sobre la renovación de los activos.

Los datos de los activos fijos se integran con la contabilidad, los cobros y pagos a fin de ofrecer una vista global y en tiempo real del estado financiero.

▶		RATIO DE ROTACION DE ACTIVOS FIJOS (VALOR OPTIMO EL MAYOR POSIBLE)
	1	VENTAS
	2	ACTIVO FIJO
1.A		A) ACCIONISTAS (SOCIOS) POR DESEMBOLSOS NO EXIGIDOS
1.B		B) INMOVILIZADO
1.B.I		I. Gastos de Establecimiento
1.B.II		II. Inmovilizaciones inmateriales
1.B.III		III. Inmovilizaciones materiales
1.B.IV		IV. Inmovilizaciones financieras
1.B.V		V. Acciones Propias
1.B.VI		VI. Deudores por operaciones de tráfico a largo plazo
1.C		C) GASTOS A DISTRIBUIR EN VARIOS EJERCICIOS

Ilustración 4 Activos Fijos

Contabilidad

La contabilidad se define como un sistema adaptado para clasificar los hechos económicos que ocurren en el negocio. De tal manera que constituya el eje central para llevar a cabo los diversos procedimientos que conducirán a la obtención del máximo rendimiento económico que implica el constituir una empresa determinada. Su primordial objetivo es suministrar información razonada, con base en registros técnicos, de las operaciones realizadas por un ente privado o público.

Las principales ventajas que se tienen en Navision con respecto a la contabilidad son las siguientes:

- Optimizar las prácticas rutinarias de contabilidad con procesos financieros automatizados y una interfaz intuitiva.
- Supervisar el rendimiento, identificar las tendencias financieras y registrar problemas potenciales.
- Satisfacer los requisitos de contabilidad internacionales más exigentes para dirigir el negocio de forma global.
- Centralizar todos los datos financieros principales incluidos los datos reales, el presupuesto, la divisa extranjera y los saldos promedio en una única contabilidad.

Gracias a esas ventajas, se pueden realizar las siguientes funciones:

- Definir múltiples relaciones entre la empresa matriz y la empresa colaboradora. Gestionar movimientos aislados o periódicos y distribuir los costes de los pedidos de ventas.
- Crear presupuestos con un análisis de cada cuenta de contabilidad y para cada cuenta total en el plan de cuentas, como se muestran en las

ilustraciones 5, 6, 7 donde se definen las diferentes cuentas dentro de un grupo principal.

- Asignar información financiera que identifique características o dimensiones como el producto, el área de ventas o el periodo de tiempo para analizar mejor el rendimiento. Asignar varias dimensiones a cada movimiento del presupuesto y filtrar el presupuesto por un máximo de cuatro dimensiones.
- Utilizar las dimensiones para crear previsiones de cuentas complejas para poder mejorar la creación de informes con opciones avanzadas.
- Simplificar la gestión de movimientos con movimientos generales, periódicos y que se reviertan de forma automática, como se muestra en las ilustraciones 8 y 9(a pesar de que ponga IVA16 en la descripción, la configuración es con el nuevo IVA), donde se observan los diferentes asientos que se pueden realizar mediante los diarios. Registrar movimientos frecuentes en la contabilidad mediante un diario periódico.
- Crear tantos diarios como desee el cliente, cada uno con sus propias series de números de documentos.
- Comprobar los saldos del diario antes de que se registren los movimientos. Consultar cómo afectarán los movimientos a las cuentas de liquidez para poder realizar los ajustes necesarios.
- Definir la fecha del año fiscal de la empresa y especificar los periodos contables que la empresa utilizará con los presupuestos, estadísticas e informes.

Nº fila	Descripción	T..	Sumatorio	M..
	ANALISIS LIQUIDEZ	C..		Si
		C..		Si
	Activo circulante	C..		Si
101	Existencias	C..	3	Si
102	Clientes	C..	43|44	Si
103	Realizable	C..	50|53|54	Si
104	Disponible	C..	57	Si
	Reservas acciones propias	C..		Si
105	Activo circulante, total	F..	101..104	Si
		C..		Si
	Pasivo a corto	C..		Si
111	Créditos	C..	51|52|56|58|59	Si
112	Proveedores	C..	40	Si
113	H.P.	C..	47	Si
114	Personal	C..	46	Si
115	Otras deudas	C..	41	Si
116	Pasivo a corto, total	F..	111..115	Si
		C..		Si
	Activo circulante menos pasivo a corto	F..	105|116	Si

Ilustración 5 Análisis de Cuentas

Nº fila	Descripción	T..	Sumatorio	M..
1.A	A) ACCIONISTAS (SOCIOS) POR DESEMBOLSOS NO EXIGIDOS	Fó	0|191|192|193|194|195|196	Si
1.B	B) INMOVILIZADO	F..	1.B.I+1.B.II+1.B.III+1.B.IV+...	Si
1.B.I	I. Gastos de Establecimiento	C..	20	Si
1.B.II	II. Inmovilizaciones inmateriales	C..	21|281|291	Si
1.B.III	III. Inmovilizaciones materiales	C..	22|23|282|292	Si
1.B.IV	IV. Inmovilizaciones financieras	C..	240|241|242|243|244|245|24...	Si
1.B.V	V. Acciones Propias	C..	198	Si
1.B.VI	VI. Deudores por operaciones de tráfico a largo plazo	C..		Si
1.C	C) GASTOS A DISTRIBUIR EN VARIOS EJERCICIOS	C..	27	Si
1.D	D) ACTIVO CIRCULANTE	F..	1.D.I+1.D.II+1.D.III+1.D.IV...	Si
1.D.I	I. Accionistas por desembolsos exigidos	C..	558	Si
1.D.II	II. Existencias	C..	30|31|32|33|34|35|36|39	Si
AUX1		C..	430|431|432|433|435|436|44...	No
AUX2		C..	551|552|553	No
1.D.III	III. Deudores	F..	AUX1[AUX2	Si
1.D.IV	IV. Inversiones financieras temporales	C..	53|540|541|542|543|545|546...	Si
1.D.V	V. Acciones propias a corto plazo	C..		Si
1.D.VI	VI. Tesorería	C..	57	Si
1.D.VII	VII. Ajustes por periodificación	C..	480|580	Si
	TOTAL GENERAL (A+B+C+D)	F..	1.A+1.B+1.C+1.D	Si
2.A	A) FONDOS PROPIOS	F..	2.A.I+2.A.II+2.A.III+2.A.IV...	Si
2.A.I	I. Capital suscrito	C..	10	Si
2.A.II	II. Prima de emisión	C..	110	Si
2.A.III	III. Reserva de revalorización	C..	111	Si
2.A.IV	IV. Reservas	C..	112|113|114|115|116|117|118	Si
2.A.V	V. Resultados de ejercicios anteriores	C..	120|121|122	Si
2.A.VI	VI. Pérdidas y Ganancias (bfº o pérdida)	C..	129|6|7	Si
2.A.VII	VII. Dividendo a cuenta entregado en el ejercicio	C..	557	Si
2.A.VIII	VIII.Acciones propias para reducción de capital	C..	199	Si
2.B	B) INGRESOS A DISTRIBUIR EN VARIOS EJERCICIOS	C..	13	Si
2.C	C) PROVISIONES PARA RIESGOS Y GASTOS	C..	14	Si
2.D	D) ACREEDORES A LARGO PLAZO	C..	15|16|17|18|248|249|259|4791	Si
AUX3		C..	400|401|402|403|406|41|437...	No

Ilustración 6 Balance de Situación

Nº fila	Descripción	T..	Sumatorio	M..
	A) GASTOS(A.1 a A.15)	Fó	1+A.2+A.3+A.4+A.5+A.6...	Si
A.1	A.1. Consumos de explotación	C..	60|61|71	Si
A.2	A.2. Gastos de personal	F..	A.2.A+A.2.B	Si
A.2.A	a) Sueldos, salarios y asimilados	C..	640|641	Si
A.2.B	b) Cargas sociales	C..	642|643|649	Si
A.3	A.3. Dotaciones para amortizaciones de inmovilizado	C..	68	Si
A.4	A.4. Variación de las provisiones de tráfico y pérdidas de créditos incobrables	C..	650|693|694|695|793|794|795	Si
A.5	A.5. Otros gastos de explotación	C..	62|631|634|636|639|651|659...	Si
A.I	A.I. BENEFICIOS DE EXPLOTACION(B.1-A.1-A.2-A.3-A.4-A.5)	F..	B.1-A.1-A.2-A.3-A.4-A.5	Si
A.6	A.6. Gastos financieros y gastos asimilados	F..	A.6.A+A.6.B+A.6.C+A.6.D	Si
A.6.A	a) Por deudas con empresas del grupo	C..	6610|6615|6620|6630|6640|6...	Si
A.6.B	b) Por deudas con empresas asociadas	C..	6611|6616|6621|6631|6641|6...	Si
A.6.C	c) Por otras deudas	C..	6613|6618|6622|6623|6632|6...	Si
A.6.D	d) Pérdidas de inversiones financieras	C..	666|667	Si
A.7	A.7. Variación de las provisiones de inversiones financieras	C..	6963|6965|6966|697|698|699...	Si
A.8	A.8. Diferencias negativas de cambio	C..	668	Si
A.II	A.II. RESULTADOS FINANCIEROS POSITIVOS (B.2+B.3-A.6-A.7-A.8)	F..	B.2+B.3-A.6-A.7-A.8	Si
A.III	A.III. BENEFICIOS DE LAS ACTIVIDADES ORDINARIAS (A.I + A.II - B.I. - B.II)	F..	A.I+A.II-B.I-B.II	Si
A.9	A.9. Variación de las provisiones de inmovilizado inmaterial, material y cartera de control	C..	691|692|6960|6961|791|792|...	Si
A.10	A.10. Pérdidas procedentes del inmovilizado inmaterial, material y cartera de control	C..	670|671|672|673|676	Si
A.11	A.11. Pérdidas por operaciones con acciones y obligaciones propias	C..	674	Si
A.12	A.12. Gastos extraordinarios	C..	678	Si
A.13	A.13. Gastos y pérdidas de otros ejercicios	C..	679	Si
A.IV	A.IV. RESULTADOS EXTRAORDINARIOS POSITIVOS (B.4+B.5+B.6+B.7+B.8-A.9-A.10-A.11-A.12-A.13)	F..	B.4+B.5+B.6+B.7+B.8-A.9-A...	Si
A.V	A.V. BENEFICIOS ANTES DE IMPUESTOS(A.III+A.IV-B.III-B.IV)	F..	A.III+A.IV-B.III-B.IV	Si
A.14	A.14. Impuesto sobre sociedades	C..	630|6323|6328|633|638	Si
A.15	A.15. Otros impuestos	C..	6320|635	Si
A.VI	A.VI. RESULTADO DEL EJERCICIO (BENEFICIOS)(A.V-A.14-A.15)	F..	A.V-A.14-A.15	Si
	B) INGRESOS(B.1 a B.8)	F..	B.1+B.2+B.3+B.4+B.5+B.6+...	Si
B.1	B.1. Ingresos de explotación	F..	B.1.A+B.1.B	Si
B.1.A	a) Importe neto de la cifra de negocios	C..	70	Si
B.1.B	b) Otros ingresos de explotación	C..	73|74|75|790	Si
B.I	B.I. PERDIDAS DE EXPLOTACION(A.1+A.2+A.3+A.4+A.5-B.1)	F..	A.1+A.2+A.3+A.4+A.5-B.1	Si

Ilustración 7 Pérdidas y Ganancias

Nº asiento	134806										
23/12/10	Fa	FC7-07/0096	6280006	Internet	Factura FRAC-07/0006748	Co	NA	VA	84,90		
23/12/10	Fa	FC7-07/0096	4720016	IVA soportado al 16 %	Factura FRAC-07/0006748				13,58		
23/12/10	Fa	FC7-07/0096	6280003	Teléfono	Factura FRAC-07/0006748	Co	NA	VA	18,75		
23/12/10	Fa	FC7-07/0096	4720016	IVA soportado al 16 %	Factura FRAC-07/0006748				3,00		
23/12/10	Fa	FC7-07/0096	4100020	Acreedores Suministros	Factura FRAC-07/0006748					120,23	
Nº asiento	134825										
23/12/10	Fa	FC8-08/0000	6280006	Internet	Factura FRAC-08/0000005	Co	NA	VA	71,90		
23/12/10	Fa	FC8-08/0000	4720016	IVA soportado al 16 %	Factura FRAC-08/0000005				11,50		
23/12/10	Fa	FC8-08/0000	6280003	Teléfono	Factura FRAC-08/0000005	Co	NA	VA	17,69		
23/12/10	Fa	FC8-08/0000	4720016	IVA soportado al 16 %	Factura FRAC-08/0000005				2,83		
23/12/10	Fa	FC8-08/0000	4100020	Acreedores Suministros	Factura FRAC-08/0000005					103,92	
Nº asiento	134826										
23/12/10	Fa	FC8-08/0000	6280006	Internet	Factura FRAC-08/0000006	Co	NA	SE	173,57		
23/12/10	Fa	FC8-08/0000	4720016	IVA soportado al 16 %	Factura FRAC-08/0000006				27,78		
23/12/10	Fa	FC8-08/0000	6280003	Teléfono	Factura FRAC-08/0000006	Co	NA	SE	17,09		
23/12/10	Fa	FC8-08/0000	4720016	IVA soportado al 16 %	Factura FRAC-08/0000006				2,73		
23/12/10	Fa	FC8-08/0000	4100020	Acreedores Suministros	Factura FRAC-08/0000006					221,17	

Ilustración 8 Asientos de Contabilidad

Fecha registro	Tipo	Nº documento	Nº cuenta	Nombre	Descripción	Tipo IVA	Grupo con	Grupo con	Debe	Haber
23/12/10	Fa	FC8-08/0000	4720016	IVA soportado al 16 %	Factura FRAC-08/0000008				3,27	
23/12/10	Fa	FC8-08/0000	4100020	Acreedores Suministros	Factura FRAC-08/0000008					23,70
Nº asiento	134829									
23/12/10	Fa	FC8-08/0000	6280006	Internet	Factura FRAC-08/0000009	C	NA	SE	171,90	
23/12/10	Fa	FC8-08/0000	4720016	IVA soportado al 16 %	Factura FRAC-08/0000009				27,51	
23/12/10	Fa	FC8-08/0000	6280003	Teléfono	Factura FRAC-08/0000009	C	NA	SE	20,20	
23/12/10	Fa	FC8-08/0000	4720016	IVA soportado al 16 %	Factura FRAC-08/0000009				3,23	
23/12/10	Fa	FC8-08/0000	4100020	Acreedores Suministros	Factura FRAC-08/0000009					222,84
Nº asiento	134830									
23/12/10	Fa	FC8-08/0000	6280003	Teléfono	Factura FRAC-08/0000010	C	NA	SE	33,64	
23/12/10	Fa	FC8-08/0000	4720016	IVA soportado al 16 %	Factura FRAC-08/0000010				5,38	
23/12/10	Fa	FC8-08/0000	4100020	Acreedores Suministros	Factura FRAC-08/0000010					39,02

Ilustración 9 Asientos de Contabilidad

Multidivisas

Las principales ventajas que se tiene en Navision con respecto a las multidivisas son las siguientes:

- Gestionar desde la facturación hasta los pagos de las cuentas empresariales en la divisa que prefieran los clientes y los proveedores.
- Optimizar los procesos de gestión de operaciones contables relacionadas con varias divisas.
- Mantener varias divisas para los cobros, los pagos, los informes de contabilidad, las cuentas bancarias y el inventario.
- Especificar métodos de conversión para los resultados en divisa extranjera a fin de simplificar el cumplimiento con las directrices de Consejo de normas financieras.
- Ofrecer a las empresas colaboradoras y otros participantes vistas de los saldos y opciones para imprimir informes financieros.

Gracias a estas ventajas, se pueden realizar las siguientes funciones:

- Definir un número ilimitado de divisas y cambios de divisa por divisa, cómo se observa en la ilustración 10, donde se muestran las diferentes definiciones de divisas.
- Mostrar la actividad de los informes financieros en varias divisas o convertir informes existentes a varias divisas.
- Registrar de forma automática las pérdidas y ganancias de las ventas y las compras producidas por los cambios de divisa
- Establecer cambios de divisa relacionales en el formato aplicable a la empresa. Definir cambios de divisa específicos para los movimientos diarios y para el ajuste de las cuentas de balance en un periodo específico.
- Especificar una divisa predeterminada para cada cuenta de cliente o de proveedor y administrar toda la cuenta en esa divisa.
- Crear y distribuir transacciones entre empresas que acepten varias divisas. Permitir que las empresas colaboradoras y los participantes globales consulten saldos e impriman informes financieros.

	Código	Descripción	D..	Cta. dif. ...	Cta. dif. ...	Cta. dif. ...	Cta. dif. ...	Prec. re...	Nº decim...	Precisión...	T..	Prec. re...	Nº decim...	Precisión...
	ATS	Chelín austriaco		7681001	6681001	7681001	6681001	0,01	2:2	0,01	M..	0,01	2:5	0,00
	BEF	Franco belga		7681001	6681001	7681001	6681001	1,00	0:0	1	M..	0,01	0:3	0
▶	CHF	Franco suizo		7681001	6681001	7681001	6681001	0,01	2:2	0,01	M..	0,01	2:5	0,00
	CZK	Corona checa		7681001	6681001	7681001	6681001	0,01	2:2	0,01	M..	0,01	2:5	0,00
	DEM	Marco alemán	✔	7681001	6681001	7681001	6681001	0,01	2:2	0,01	M..	0,01	2:5	0,00
	DKK	Corona danesa		7681001	6681001	7681001	6681001	0,01	2:2	0,01	M..	0,01	2:5	0,00
	ESP	Peseta española	✔	7681001	6681001	7681001	6681001	0,01	2:2	0,01	M..	0,01	2:5	0,00
	FRF	Franco francés	✔	7681001	6681001	7681001	6681001	0,01	2:2	0,01	M..	0,01	2:5	0,00
	GBP	Libra británica		7681001	6681001	7681001	6681001	0,01	2:2	0,01	M..	0,01	2:5	0,00
	ISK	Corona islandesa		7681001	6681001	7681001	6681001	0,01	2:2	0,01	M..	0,01	2:5	0,00
	NLG	Guilder holandés	✔	7681001	6681001	7681001	6681001	0,01	2:2	0,01	M..	0,01	2:5	0,00
	NOK	Corona noruega		7681001	6681001	7681001	6681001	0,01	2:2	0,01	M..	0,01	2:5	0,00
	SEK	Corona sueca		7681001	6681001	7681001	6681001	0,01	2:2	0,01	M..	0,01	2:5	0,00
	SIT	Tólar esloveno		7681001	6681001	7681001	6681001	0,01	2:2	0,50	M..	0,01	2:5	0,00
	USD	US-Dólar		7681001	6681001	7681001	6681001	0,01	2:2	0,01	M..	0,01	2:5	0,00

Ilustración 10 Definición de Divisas

Cobros

Las principales ventajas que se tienen en Navision con respecto a los cobros son las siguientes:

- Realizar el seguimiento de facturas, pagos parciales y automatizar los procesos de aprobación de los documentos.
- Crear resultados personalizados, ampliar el crédito a los clientes y ofrecer descuentos destinados a clientes preferenciales, todos basados en los análisis realizados.
- Aumentar la eficacia con una interfaz gráfica fácil de utilizar para que los profesionales de finanzas tengan un acceso rápido a la información.
- Hacer uso de la cuenta de cliente y el seguimiento de la eficacia de ventas para ayudar a identificar tendencias, planificar estrategias de ventas y administrar cuentas.

Gracias a estas ventajas se pueden realizar las siguientes funciones:

- Seleccionar la forma de aplicar los cobros de los clientes a las facturas.
- Crear previsiones que calculen el interés, amorticen las cantidades y creen avisos para los clientes con saldos vencidos.
- Definir un número ilimitado de términos de cargos financieros incluidos los tipos de interés, los periodos de gracia, los importes mínimos y la divisa y crear abonos de cargo financieros de forma manual o automática basándose en fechas de vencimiento preestablecidas.
- Realizar el seguimiento de una factura prepago en un pedido de ventas hasta que el pedido esté facturado por completo. Una vez facturado, se

debe registrar el cobro de dicha factura mediante el diario de cobros como se observa en la ilustración 11.

- Automatizar las ventas estándar o repetitivas estableciendo líneas y descuentos de documentos de ventas.
- Definir una gran variedad de términos de descuentos, incluidos los descuentos basados en un porcentaje de la venta global para poder aplicarlos a líneas de pedidos individuales.
- Establecer una ficha de cuenta para cada uno de los clientes que especifiquen los términos de pago como la divisa, la información sobre impuestos y las fechas de vencimiento de pago, como se observa en la ilustración 12, donde se define cada uno de esos datos.
- Establecer un nivel de tolerancia que permita cerrar por completo una factura y movimiento de cobro incluso cuando la factura no se haya pagado por completo
- Utilizar los grupos de registro para calcular de forma automática los detalles del impuesto sobre el valor añadido (IVA) y el impuesto de ventas de cada uno de los clientes de modo que se pueda enviar fácilmente esta información a las autoridades relevantes.

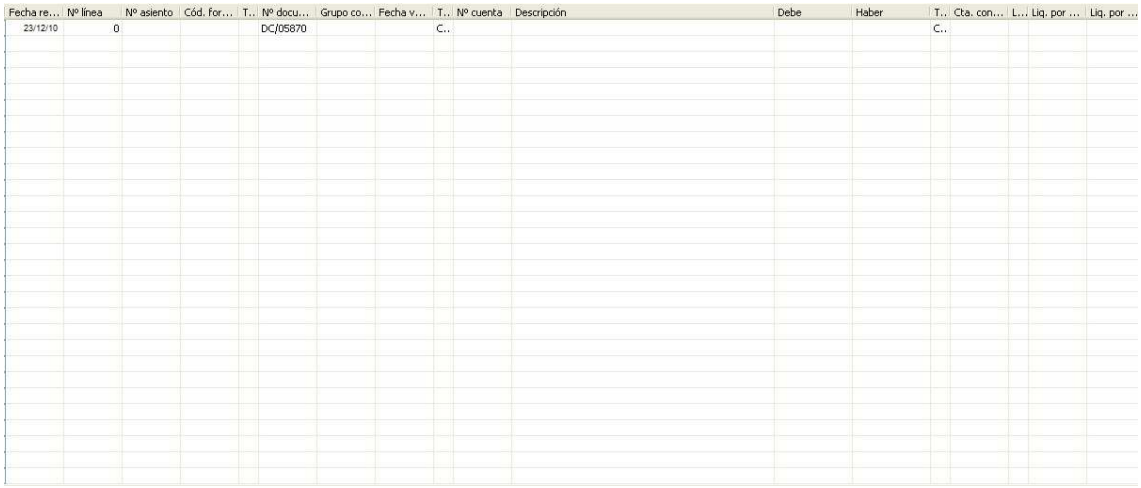


Ilustración 11 Diario de Cobros

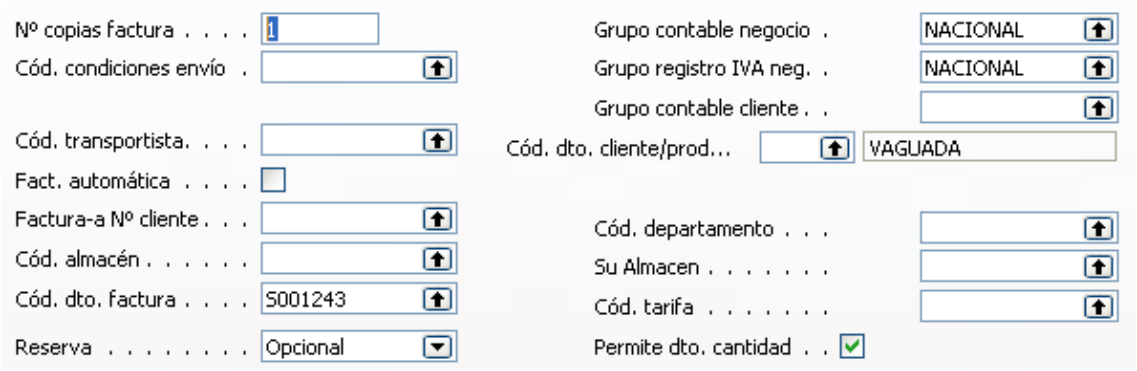


Ilustración 12 Condiciones de Cobro de Clientes

Pagos

Las principales ventajas en Navision con respecto a los pagos son las siguientes:

- Gestionar los pagos de la manera más eficaz con la planificación automatizada de pagos, las aprobaciones de documentos y la información en tiempo real.
- Negociar mejores términos y contratos gracias al acceso a saldos de cuentas.
- Aplazar pagos o costes durante varios periodos con los cálculos gestionados automáticamente.
- Una interfaz gráfica fácil de utilizar ayuda a que los profesionales de las finanzas tengan acceso rápido a la información necesaria y realicen análisis para obtener detalles del pago.

Gracias a estas ventajas se pueden realizar las siguientes funciones:

- Registrar e imprimir facturas de proveedores y abonos con la facturación de compras. Los movimientos con proveedores se registran automáticamente.
- Distribuir la factura de proveedores a filiales con procesos de pagos de cuentas.
- Optimizar procesos mediante diarios periódicos para las tareas más comunes como los pagos mensuales de arrendamientos o las facturas de acceso a Internet, como se muestra en la ilustración 14.
- Definir las divisas de los movimientos comerciales y determinar la forma en que se redondean y registran esas divisas. Calcular las pérdidas y ganancias por cambio de divisa de forma automática.

- Automatizar compras repetitivas estableciendo líneas de compra estándar que se insertarán en los pedidos de compra.
- Dar prioridad al orden en el que se debe pagar a los proveedores o determinar las previsiones de pago según disponibilidad del descuento. Detener el pago de ciertos pedidos de compra hasta que se apruebe la factura de compra.
- Establecer registros de cuentas para cada uno de los proveedores en los que se especifican los términos de pago como la divisa, la información sobre impuestos y las fechas de vencimiento de pago, como se muestra en la ilustración 13.
- Calcular el impuesto sobre productos importados al adquirirlos de los proveedores y realizar un seguimiento de las ventas o impuestos sobre el valor añadido (IVA) que se abona a los proveedores.

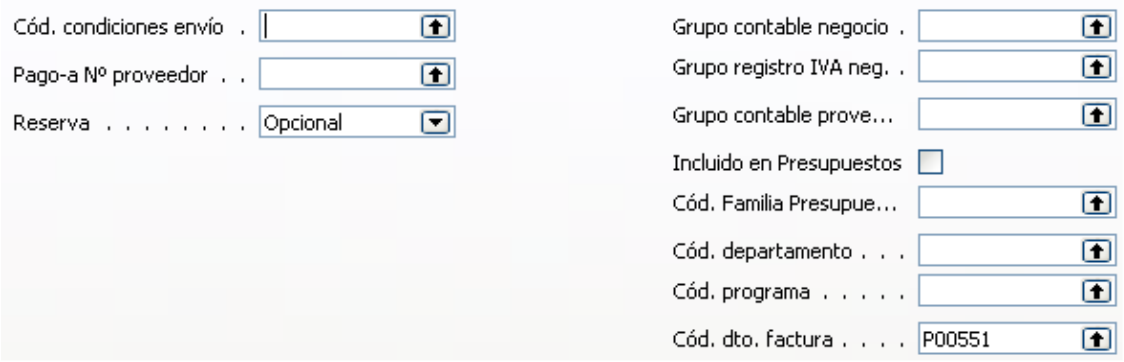


Ilustración 13 Condiciones de Pago a los Proveedores

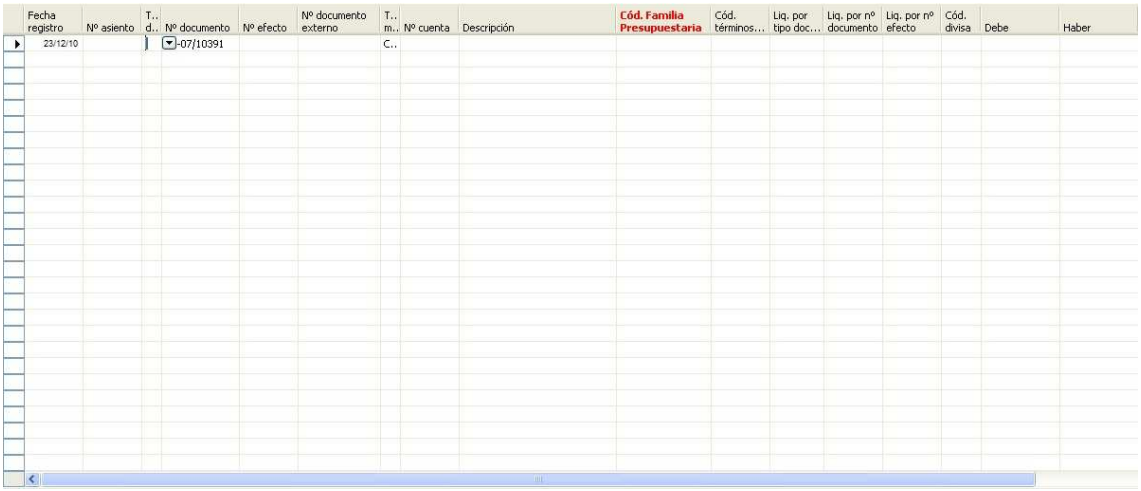


Ilustración 14 Diario de Pagos

Fabricación

Navision ofrece una familia integrada de aplicaciones de fabricación e incluye herramientas para planificar, gestionar y ejecutar operaciones de fabricación. Gestionar todo el proceso de fabricación, desde la configuración de los productos, la planificación de la capacidad y previsión de la demanda, hasta la programación de la producción en planta.

Las principales ventajas en Navision con respecto a la fabricación son las siguientes:

- Optimizar las operaciones automatizando los procesos de fabricación y obtener una mayor visibilidad de todos los aspectos de las operaciones desde la entrada del pedido hasta la producción, la gestión del almacén y la entrega.
- Responder rápidamente a las consultas de los clientes.
- Planificar pedidos urgentes y realizar cambios de última hora en los procesos de fabricación.
- Mediante un explorador WEB, los proveedores pueden administrar catálogos, escribir pedidos de envío directo y mantener las fechas de entrega.

Gracias a estas ventajas, se pueden realizar las siguientes funciones:

- Mejorar la coordinación de solicitudes de pedidos a medida y simplificar las decisiones de compra o fabricación. Modificar los componentes y las operaciones según las necesidades. Planificar pedidos de una familia de productos, como se muestra en la ilustración 15.
- Integrar diferentes tipos de listas de materiales y definiciones personalizadas en operaciones de fabricación para mejorar el rendimiento a la vez que se cumple el tiempo de lanzamiento al mercado.
- Obtener acceso a la planificación para reorganizar las operaciones.
- Los mensajes de acción, el seguimiento y las diversas opciones de planificación permiten realizar excepciones y cambios de última hora.

- Realizar las planificaciones a partir del pedido de venta, orden de producción o pedido de compra. Facilitar el flujo de materiales mediante la cadena de suministro para la planificación en varias ubicaciones.
- Analizar los patrones de ventas desde varias perspectivas. Comparar la demanda prevista con las ventas reales para realizar planes de demanda.
- Implementar planes realistas según las demandas de capacidad de recursos entrantes. Vuelve a definir las modificaciones de pedidos y las políticas de reaprovisionamiento según las necesidades.
- Integrar a la perfección la fabricación y la funcionalidad de almacén para ayudar a optimizar el diseño del almacén y la utilización del espacio.
- Conocer los costes de los productos mediante el proceso de producción para gestionar de forma eficaz los precios de venta y compra, como se muestra en la ilustración 16.

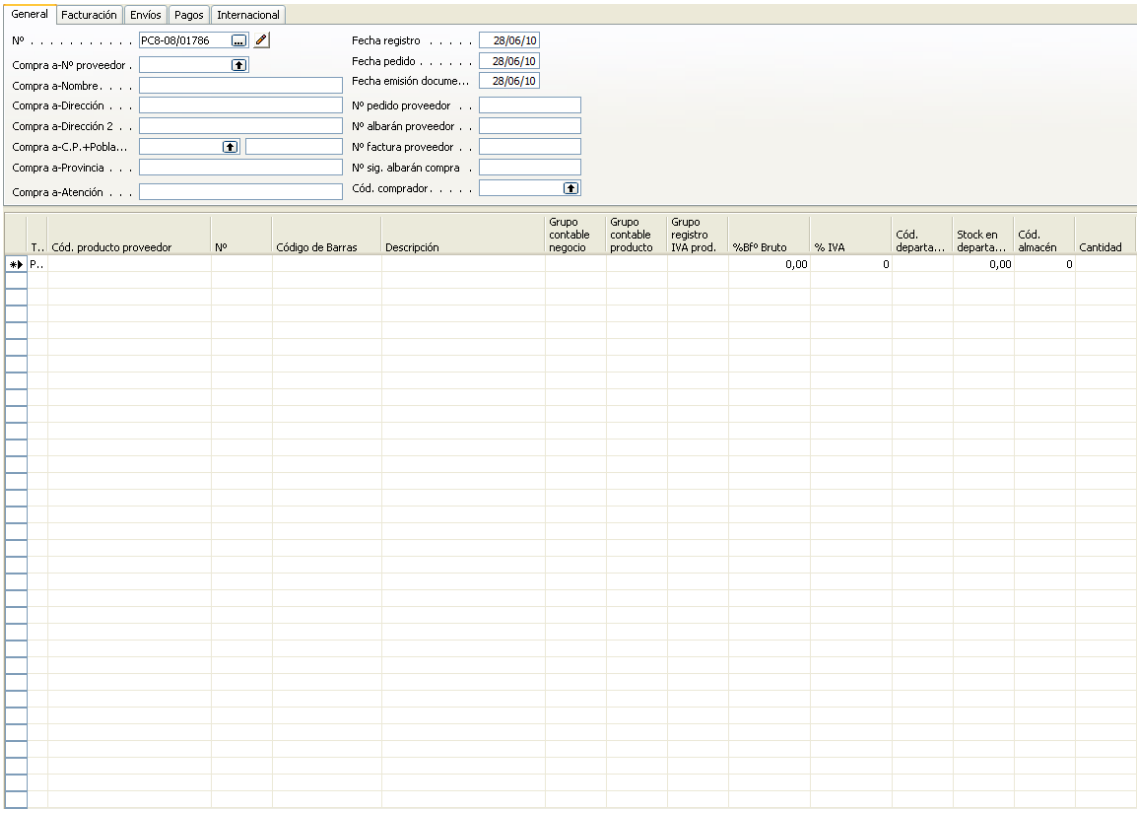


Ilustración 15 Pedido de Compra

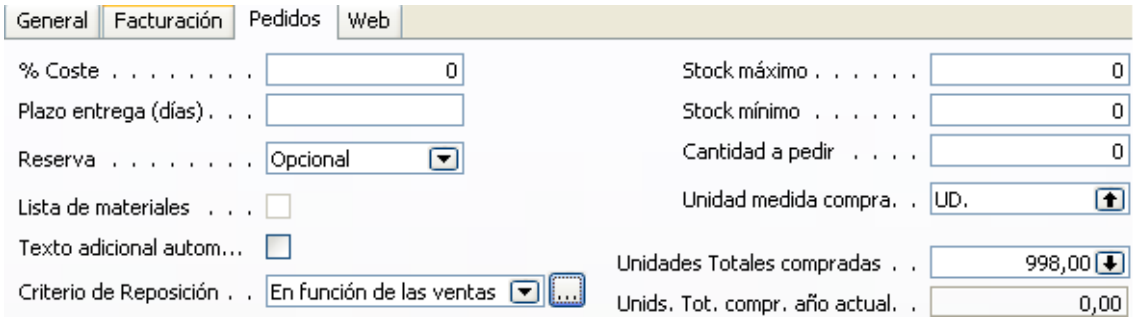


Ilustración 16 Información del Producto en los Pedidos

Planificación del suministro

Las principales ventajas en Navision con respecto a la planificación del suministro son las siguientes:

- Obtener una visión global del proceso de fabricación mientras se evite la complejidad y el gasto de toda la planificación.
- Calcular la disponibilidad de los artículos para poder cumplir sus compromisos mediante herramientas que ofrecen información en tiempo real.
- Adaptar los métodos y procesos de fabricación para reflejar los cambiantes requerimientos de los clientes.
- Utilizar diversas opciones de planificación para satisfacer la demanda a la vez que se minimiza el coste total por producción y almacenamiento de artículos.
- Tomar mejores decisiones sobre los precios mediante una replanificación simultánea de costes, materiales y operaciones.

Gracias a estas ventajas, se pueden realizar las siguientes funciones en Navision:

- Facilitar información eficaz y flujo de material a través de la cadena de suministro, para asegurar que los materiales se encuentran en la posición correcta.
- Optimizar el uso de materiales mejorando la exactitud de la previsión de la demanda. Utilizar escenarios hipotéticos para determinar la forma en que las modificaciones a los procesos de fabricación pueden cambiar los requerimientos de los materiales.
- Trabajar con un único diario de producción que combina las funciones de diario de consumo y el diario de resultados, como se muestra en la ilustración 17.

- Planificar a partir del pedido de venta, orden de producción o pedido de compra la utilización de los métodos tradicionales de la producción maestra o de planificación. Los planificadores de las compras pueden reconocer la forma de ajustar la planificación de entrega.
- Actualizar y planificar todos los materiales, los costes y las operaciones de forma simultánea si se producen cambios en los métodos de fabricación.
- Realizar el seguimiento de un lote o de una serie para determinar dónde se compraron o vendieron los productos.

La planificación a partir de los pedidos de venta otorga la flexibilidad de definir la estructura de pedidos de productos, lo que facilita realizar el seguimiento del estado de los pedidos en varias líneas o niveles, como se observa en la ilustración 18.

	Nº producto	Descripción	Viene de Almacen	Cód. almacén	Fecha registro	T.. m..	Nº documento	Cantidad	Cód. unidad medida	Precio unitario	Importe	Precio coste
✱				0	28/06/10	E..	DGP-07/...	1		0,00	0,00	0,00

Ilustración 17 Diario de Productos

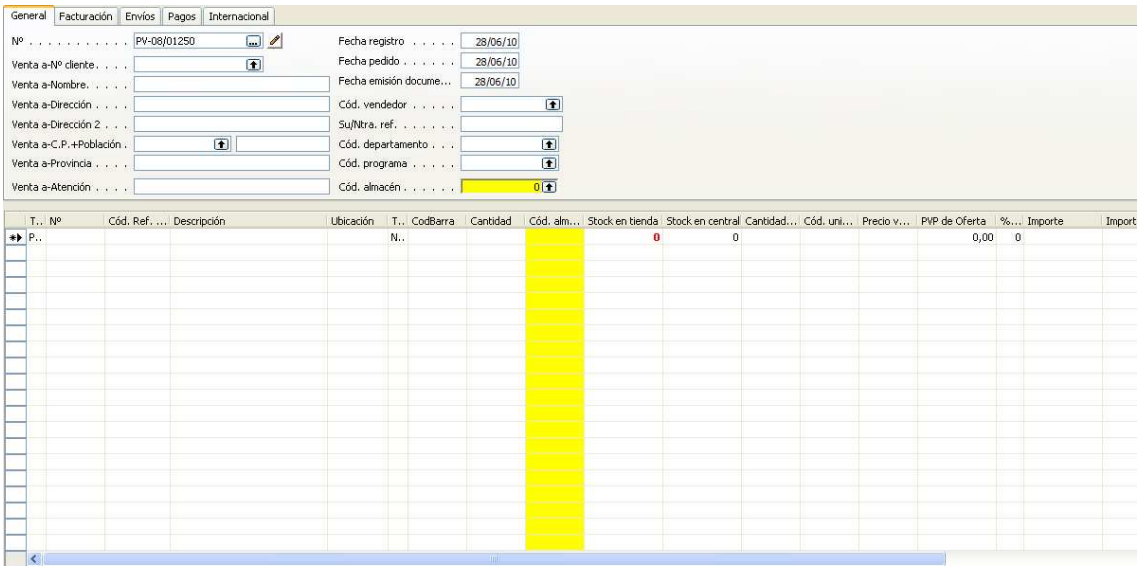


Ilustración 18 Pedido de Venta

Gestión de la Cadena de Suministro

Racionalizar ventas, compras y ciclos de picking. Con Navision, se puede personalizar el flujo de trabajo de procesos para satisfacer las necesidades específicas y seguir el ritmo de competitividad del mercado y los márgenes reducidos. Incrementar la fidelidad de los clientes ofreciéndoles una mejor respuesta. Aprovechar las oportunidades de mercado para mejorar la rentabilidad trabajando más eficazmente con el partner.

Las principales ventajas que se tienen en Navision con respecto a la cadena de suministro son las siguientes:

- Establecer criterios que guíen los patrones de recogida, así como las cantidades y los lugares de las ubicaciones.
- Reducir varias gestiones y cuellos de botella, mediante picking y ubicación controlados. Especificar información de gestión del material directamente desde el almacén para aumentar la eficacia.
- Actualizar automáticamente el estado del empaquetado para asegurar el cumplimiento adecuado del pedido.
- Seleccionar el nivel de sofisticación necesario con la confianza de que la solución puede adaptarse fácilmente al crecimiento del negocio y a nuevos procesos.

Gracias a estas ventajas, se pueden realizar las siguientes funciones en Navision:

- Integrar el procesamiento de pedidos, la fabricación y la funcionalidad del almacén para optimizar la utilización del diseño y el espacio, administrar la reposición y gestionar varios pedidos a la vez, como se muestra en la ilustración 19. Incorporar una gran variedad de métodos de priorización de picking, incluidos los métodos FIFO, FEFO o LILO.
- Realizar el picking o la ubicación de productos y registros de inventario independientemente de documentos de origen cómo recepciones de compra o documentos de ventas para mantener el inventario exacto.
- Aumentar la precisión y la eficacia de la administración del almacén realizando picking y ubicación de elementos, recuentos de inventarios físicos y desplazando productos.
- Realizar el seguimiento de números de lote o de serie para determinar rápidamente dónde se compraron, procesaron o vendieron los productos, como se muestra en la ilustración 20.
- Conocer los costes de los productos en todo proceso de producción incluido el inventario, el trabajo en proceso y el coste de bienes vendidos.
- Controlar la distribución relacionando los transportistas con los servicios que ofrecen.
- Organizar de forma automática abonos, productos de reposición, devoluciones a proveedores o la devolución parcial o combinada de envíos o recepciones.
- Determinar la frecuencia del recuento cíclico por producto o unidad de almacenamiento para ayudar a aumentar la exactitud de inventario y cumplir los plazos de envío.

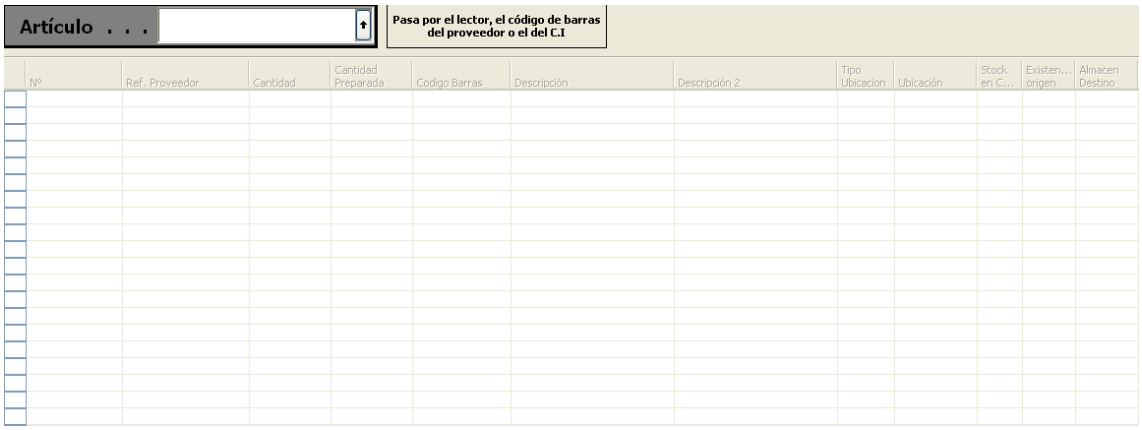


Ilustración 19 Pedido de Reposición

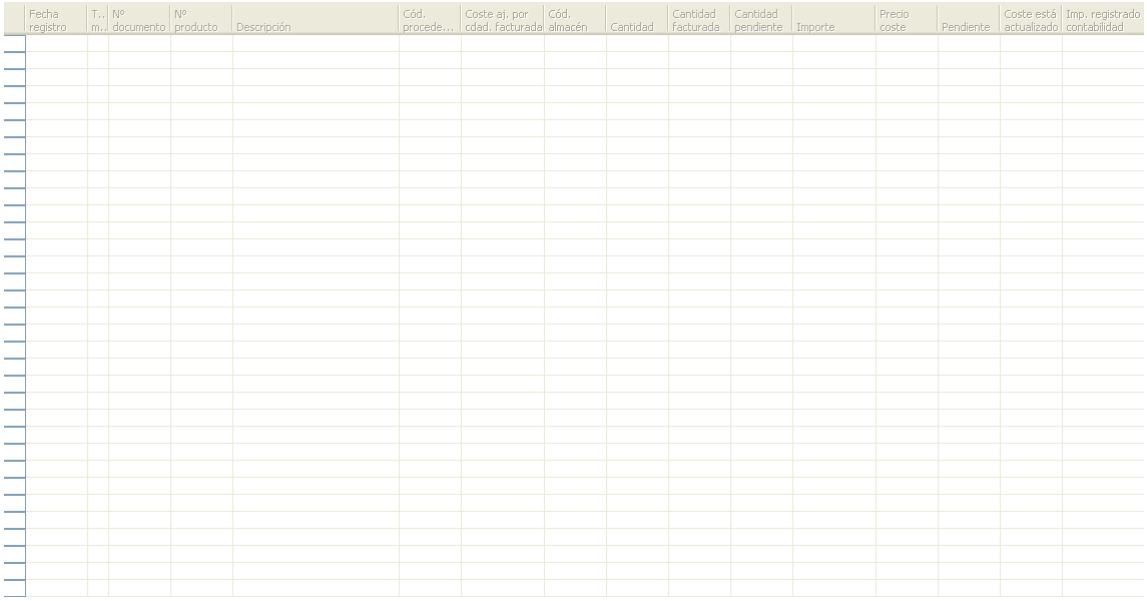


Ilustración 20 Movimientos de Producto

Inteligencia de Negocio

Aportar una visión estratégica a los procesos de negocio con sofisticadas soluciones de informes, análisis y presupuestos ayudan a mejorar y promover la toma de decisiones críticas en la organización. El acceso directo a la información crítica de negocio en tiempo real y una amplia gama de herramientas de análisis e informes ayudan a gestionar presupuestos, crear y consolidar informes y descubrir tendencias y relaciones.

Las principales ventajas que se tienen en Navision con respecto a la Inteligencia de Negocio son las siguientes:

- Los empleados pueden tener acceso a información comercial en un tiempo real, así como realizar un análisis detallado de los movimientos, los registros de clientes y los históricos, además de controlar el ritmo de las operaciones comerciales.
- Mediante acceso a información actualizada y las eficaces herramientas de análisis y creación de informes, los empleados pueden supervisar el rendimiento, analizar tendencias y descubrir problemas potenciales.
- La perfecta integración de Navision con los programas conocidos como Excel facilita tener acceso, analizar y compartir información.

Gracias a estas ventajas, se pueden realizar las siguientes funciones en Navision:

- Trabajar con rapidez gracias a las avanzadas herramientas de análisis con las que se obtendrá una vista completa del rendimiento empresarial. Establecer y revisar las muestras y los informes gráficos.
- Consultar y analizar las tendencias de ventas y rentabilidad, el movimiento de inventario, los pedidos y compromisos, las campañas de marketing y mucho más, como se muestra en la ilustración 22.
- Extraer datos de toda la empresa a la vez que se trabaja con programas de la familia Office. Asimismo, se pueden exportar datos en cualquier formato mediante los formatos de XML.
- Ofrecer acceso en línea de datos, informes y herramientas de colaboración en tiempo real con el Portal de Navision.
- Crear informes desde cero mediante asistentes de informes paso a paso, como se muestra en la ilustración 21.
- Definir y asignar características a la información registrada en el trabajo diario. Establecer dimensiones y jerarquías de valor que reflejen las necesidades de la creación de informes.
- Vincular los registros de Navision con información externa y almacenar datos en el servidor SharePoint para obtener el acceso a ellos.
- Optimizar el desarrollo gracias a un entorno de desarrollo integrado que permite satisfacer los requisitos específicos de la empresa y las necesidades cambiantes.

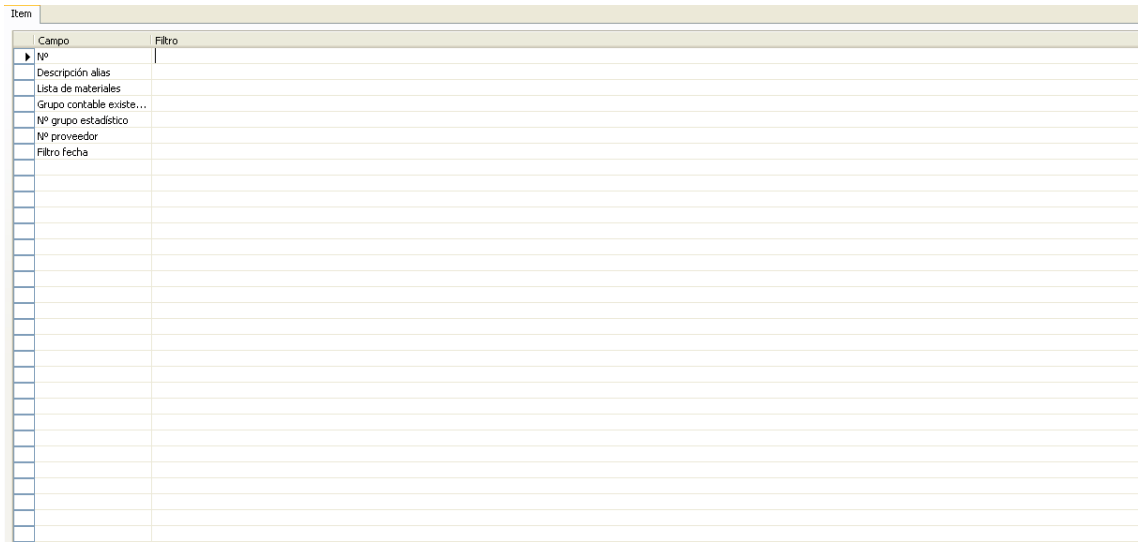


Ilustración 21 Opciones de un Informe

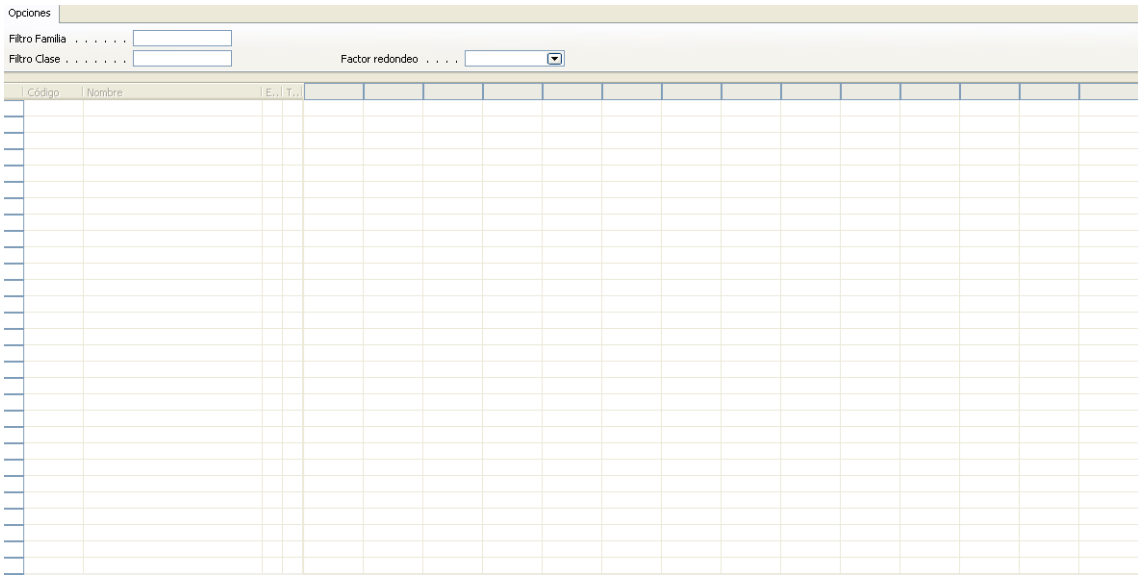


Ilustración 22 Informe de Ventas

Clientes

La gestión de relaciones con los clientes permite la automatización de muchas de las tareas diarias de los profesionales de ventas, atención al cliente y marketing, como se muestra en la ilustración 23 se pueden gestionar los registros de clientes y los históricos de ventas y realizar un seguimiento de la actividad de los clientes.

Aumentar la rentabilidad de las operaciones organizando los recursos de servicios para una óptima eficacia, previendo y realizando un seguimiento del consumo de piezas y obteniendo un control más estrecho sobre los costes.

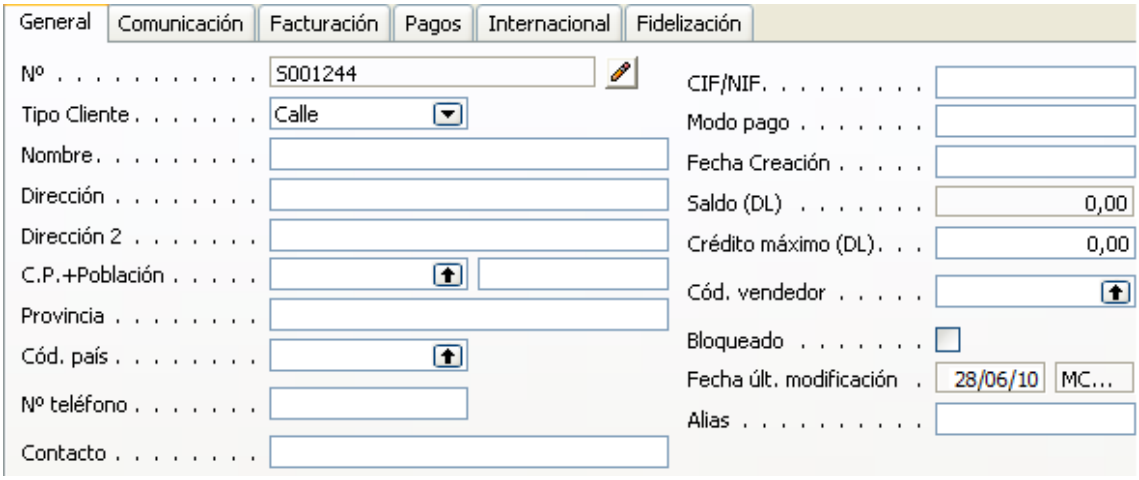


Ilustración 23 Ficha Cliente

Gestión del Servicio

Las principales ventajas en Navision con respecto a la gestión del servicio son las siguientes:

- Crear un entorno de trabajo más productivo gracias a un mayor control de los inventarios de piezas.
- Optimizar la generación, distribución, realización y facturación de pedidos de servicio y realizar el seguimiento del consumo de piezas una vez mejorado el acceso a la información actualizada, como se muestra en la ilustración 24.
- Establecer y realizar el seguimiento de garantías y contratos de licencia de servicio, así como los periodos de servicio contractuales o tiempo de respuesta de modo que los empleados puedan automatizar pedidos de servicio relacionados.

Gracias a estas ventajas, se pueden realizar las siguientes funciones:

- Crear pedidos de servicio basados en las solicitudes de cliente o los problemas post-venta. Aceptar sugerencias creadas por el sistema para abrir un pedido basado en el servicio periódico.
- Administrar contratos de licencia de servicio para anticipar necesidades de servicio, cumplir las obligaciones de los periodos de servicio o tiempos de respuesta, registrar las preferencias del cliente para los técnicos o citas de servicio y prever de forma proactiva el servicio.

- Definir el tiempo, el material y los requisitos de los recursos habituales para un tipo de servicio concreto. Dar prioridad a las tareas según sea necesario con un conocimiento claro de los pedidos de servicio abiertos, los compromisos de contrato y la carga de trabajo
- Agilizar la solución de problemas analizando anteriores servicios y proporcionando directrices y procedimientos para la futura resolución de problemas.
- Establecer unos precios de servicio incluidos unos mínimos o máximos fijos, precios específicos según cliente, diversos tipos de cargos y grupos de precios.
- Conocer el rendimiento y rentabilidad de las operaciones de servicio creando informes que muestren los pedidos de servicio abiertos.

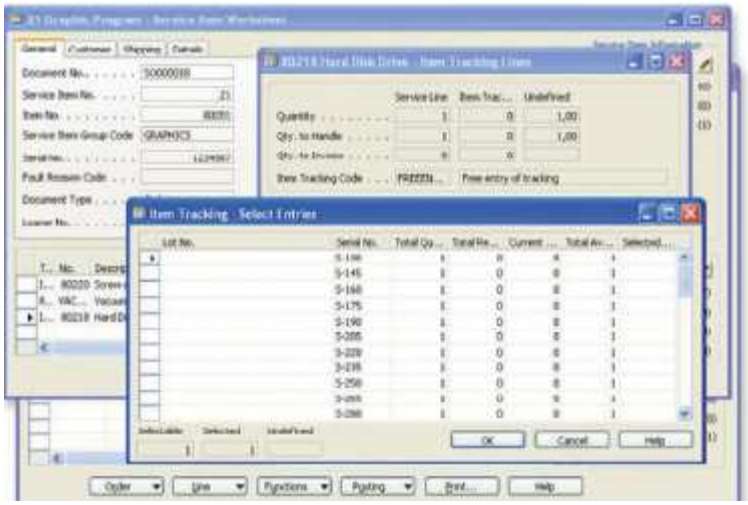


Ilustración 24 Seguimiento del Producto

[Navision] Microsoft Dynamics NAV. Disponible [Internet]: http://www.microsoft.com/dynamics/es/es/products/nav-overview.aspx

Como dato interesante se puede resaltar lo siguiente: El número de instalaciones que actualmente tiene SAP en España es de 8000, Navision consta de 2158 instalaciones y Axapta de 1300 instalaciones.

Por otro lado, cabe que destacar que el coste medio de una instalación completa en SAP es de 50.000 €, mientras que en Navision es de 30.000 € y en Axapta de 25.000€.

2.4.2 Arquitectura de Navision

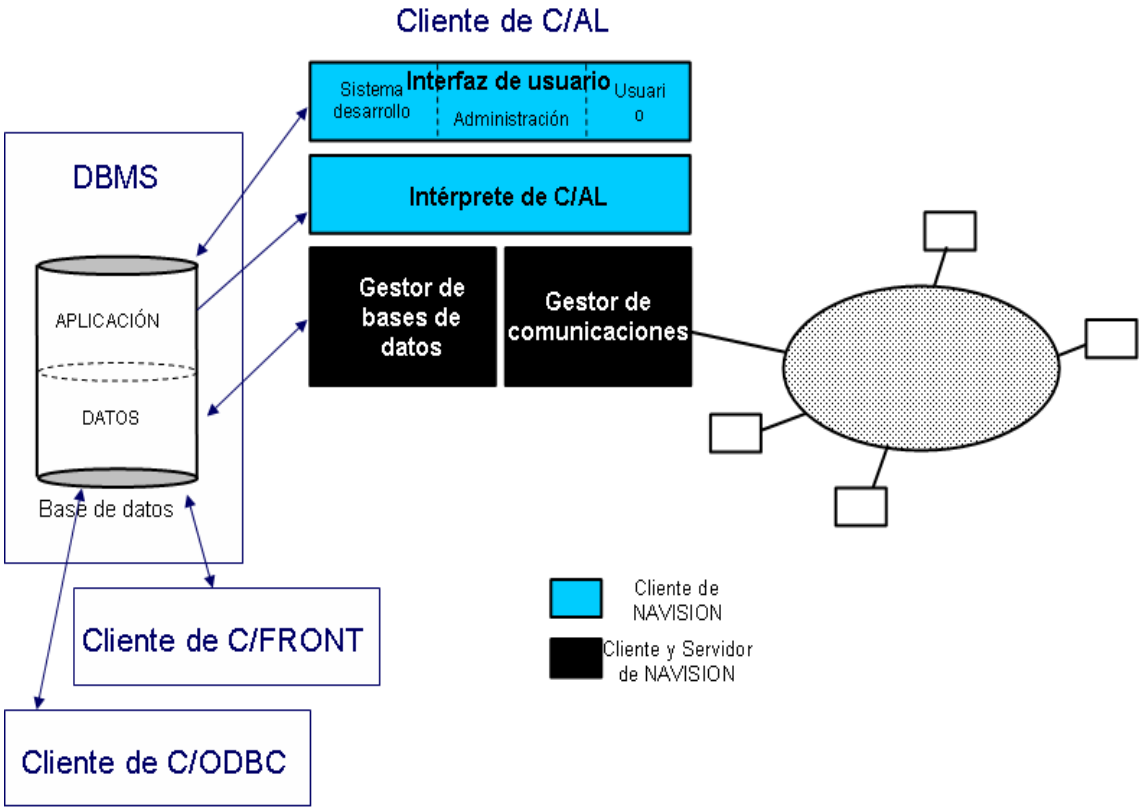


Ilustración 25 Estructura de Navision

En este apartado, se explica cómo esta estructurado Navision dentro de una empresa:

En azul, se muestra lo que puede ver cualquier persona que tenga acceso al programa de Navision y tenga instalado en la máquina el cliente de Navision, es decir, puede ver la interfaz de usuario que muestra cada uno de los módulos de que se compone la empresa, ya sea contabilidad, ventas y cobros, compras y pagos, etc. Dentro de cada módulo existen cada una de las funciones que se pueden realizar para cada departamento, en contabilidad está el plan de cuentas, el control presupuestario, etc. En ventas y cobros, están los pedidos de venta, el diario de cobros, etc. En compras y pagos, están los pedidos de compra, el diario de pagos, etc.

Cada usuario tendrá una serie de permisos, que le permiten realizar una serie de funciones dentro del programa. Si el usuario accede a una función en la que no tiene unos determinados permisos, se le emite un mensaje dónde se le dice que carece de los permisos suficientes.

En negro, se muestra todo lo que contiene el servidor de Navision, donde se encuentra la base de datos que en este caso está centralizada, lo que quiere decir que cada una de las máquinas de la empresa debe estar conectada al servidor para que pueda tener los datos en tiempo real y que las modificaciones que pueda hacer cualquier usuario se modifiquen al instante. Además, se debe instalar el cliente en el servidor para que se pueda autentificar como usuario y pueda registrar cada uno de esos movimientos en la base de datos.

3 Memoria del Trabajo Realizado

3.1 ERP para la gestión de una juguetería

En este apartado se explica, de forma genérica, cada uno de los módulos existentes en la empresa y qué funciones se pueden realizar en el Sistema ERP, que en este caso son: Contabilidad, Ventas y cobros, Compras y pagos, Almacén y Gestión de Tienda.

3.1.1 Módulo Contabilidad

En el módulo de contabilidad se realizan las siguientes funciones: creación de asientos contables, control presupuestario, configuración del plan de cuentas, configuración de una serie de aspectos de la contabilidad y control de la facturación de la empresa.

En lo referente a la creación de los asientos contables existen diferentes tipos:

- Cobros: En este caso los cobros provenientes de las ventas de las tiendas se generan de forma automática una vez que la venta se ha registrado en el sistema. Lo mismo ocurre cuando la tienda genera una reserva para el cliente. Por otro lado, los asientos que debe realizar el personal de contabilidad corresponden a la remesa de tarjetas de cada uno de los bancos con los que trabaja la empresa. En esos asientos se contempla tanto el valor de la venta que se pagó con la tarjeta cómo la comisión de la misma. En este caso, en la remesa de tarjetas se tiene en cuenta tanto la venta a clientes como la venta que se realiza en las tiendas. Por otro lado, otro tipo de asiento contable que se debe generar de forma manual son los ingresos que realizan al banco cada una de las tiendas al cierre del día. Además, otro tipo de asientos que se generan de manera manual es cuando el cliente paga la factura del pedido que se le ha servido anteriormente o cuando se le ha abonado al cliente debido a que el pedido no le llegó de forma adecuada o deteriorada.

- Pagos: En este caso todos los gastos provenientes de los gastos que realizan las tiendas se generan de forma automática en el sistema una vez se ha registrado en la tienda. Por otro lado, los asientos que debe realizar el personal de contabilidad corresponden a generar y registrar la factura de la luz, agua y gas de cada una de las tiendas. Además, otro tipo de asiento contable que se debe generar de forma manual son las devoluciones de recibos que realiza la empresa, ya que esos recibos se facturaron y para poder reflejar esa devolución del recibo se debe crear ese asiento contable. Por otro lado, cuando se recibe mercancía en el almacén, en el pedido de compra correspondiente a esos productos, se debe registrar qué se ha recepcionado de dicha mercancía. Entonces, una vez registrada provoca que esos productos estén pendientes de pagar en una serie de vencimientos (negociados con los proveedores) de pagos. Por lo tanto, cuando llega el momento de pagar, al registrar el pago se genera el asiento contable correspondiente al pago de la factura de compra. Lo mismo ocurre cuando se genera un abono de compra debido a que ha llegado producto deteriorado o han llegado cantidades inferiores a las que indica el albarán de entrega.
- Generales: En este caso, existen una serie de diarios generales para realizar las siguientes funciones: Para liquidar facturas que correspondan a publicidad proporcionada por los proveedores para colocarla en las tiendas que correspondan, para traspasos de saldos de una cuenta a otra, para anular un apunte que se haya generado de forma errónea y para regularizar las cuentas de la empresa.

Por otro lado, los responsables de esta área son los encargados del control presupuestario, es decir, son los que partiendo del presupuesto total anual deben decidir el presupuesto que destinan a cada mes, esto se debe a que en la empresa existen una serie de campañas (navidad, verano, comuniones) en las que el gasto es mucho mayor en ese periodo que en otro, por lo tanto se debe contar con ese factor a la hora de destinar presupuesto mensual. Por otro lado, dentro de cada mes también existe la posibilidad de definir qué % del presupuesto mensual se destinará a la compra de material informático, alquileres de locales, suministros, compra de productos, etc.

En lo referente al plan de cuentas, los responsables de esta área son los encargados de configurar, según necesidades de la empresa, el plan de cuentas que se quiere utilizar en la empresa. Dentro de cada cuenta, debe indicar el número de la

cuenta, descripción de la cuenta, que tipo de cuenta es, si es principal o auxiliar. En caso de ser principal, se debe indicar que cuentas auxiliares están unidas a dicha cuenta, para ello existe el campo sumatorio donde indica las cuentas dependientes de ella. Esto sirve para que a la hora de realizar compras, ventas, pagos de facturas o asientos contables, se indique a que cuenta se quiere imputar dicho movimiento. Además, para luego a final de mes realizar un balance de la empresa a partir de las cuentas que se han configurado.

Por último, el tema del control de la facturación de la empresa se refiere a que en la empresa existen empresas afiliadas a ella, que se utilizan para la facturación de las tiendas. Entonces al final de mes, la empresa crea factura por cada una de las tiendas en la cual se tiene en cuenta el producto que se le ha servido, el producto que la tienda ha devuelto y los productos que ha servido el proveedor directamente a la tienda. De cada caso, se recorren los movimientos que aún no se han procesado y se van añadiendo a una factura calculando el coste, por lo que al final te queda una factura con el importe que se le va a imputar a la empresa por todo el movimiento de producto generado.

3.1.2 Módulo de Ventas y Cobros

En este módulo se realizan las siguientes funciones: creación de pedidos de venta, facturar o abonar pedidos de venta, crear en el diario de cobros asientos para facturar las ventas que proceden de tiendas y creación de clientes.

En lo referente a la creación de pedidos se realiza de la siguiente manera:

En primer lugar, se debe indicar a que cliente va dirigido dicho pedido para que cuando se vayan insertando los productos se les aplique una serie de descuentos a tenor del tipo de cliente que sea. A continuación, se van insertando los productos en el pedido. Una vez insertado el producto se recorren las tarifas de dicho producto por si existe una tarifa especial para dicho cliente. Si no existe, le propone como precio de venta la tarifa del proveedor del producto. Una vez insertadas todas las líneas los responsables avisan al jefe de almacén de que debe preparar el pedido. Hay un caso especial de pedido: el que realizan los clientes por la Web de la empresa. En este caso la generación del pedido se realiza de forma automática y una vez generado se le envía al responsable un e-mail avisando de la creación del pedido para que lo pueda gestionar y preparar.

En lo referente a la facturación de los pedidos se realiza de la siguiente manera:

Una vez que el pedido lo haya preparado el almacén, se debe avisar al responsable de ventas que ya puede facturarlo. Lo puede hacer de dos maneras: crea una nueva factura y en ella indica que traiga todas las líneas pendientes de facturar de dicho cliente. Entonces el sistema indica en la factura a que pedido corresponden las líneas posteriores, para que pueda testificar que la facturación es correcta. La otra manera de hacer la facturación, es ir al propio pedido e indicar qué se quiere facturar.

En lo referente al abono de pedidos se puede hacer de la siguiente manera:

Se genera un nuevo abono y se van introduciendo de forma manual cada una de las líneas que se quieren abonar al cliente. Una vez acabado se registra el abono. La otra manera de realizarlos es: en los abonos existe una opción de copiar líneas de un pedido. Entonces el responsable indica a que pedido corresponde y en ese momento se añaden esas líneas al abono. Las líneas que no se quieran abonar, el responsable puede borrarlas sin problemas. Una vez hecho ese proceso, se registra el abono.

En lo referente a los diarios de cobros existen dos tipos de asientos contables referentes a ese diario:

- Automáticos: Estos asientos se generan y registran de manera automática cuando en el programa entra una venta de una tienda. En ese momento se genera un asiento en el diario de cobro para dicha venta y se registra, para poder certificar que se ha cobrado dicha venta.
- Manuales: Estos asientos los genera el personal de contabilidad y su origen puede ser por varios motivos: cuando el cliente ha pagado la factura del pedido que ha recibido, se debe generar un asiento en el diario de cobros para dejar constancia del cobro de dicha factura. Otro motivo es cuando se ha ingresado en el banco una remesa de tarjetas con las que han pagado los clientes, se debe generar un asiento en el cual se certifique el cobro de dichas tarjetas. Por otro lado, cuando se quiere regular la cuenta de un cliente, se debe generar un asiento en el diario de cobro para que quede constancia de dicha regulación. Además, cuando se hace la devolución de una factura a un cliente, se debe generar ese asiento contable para dejar constancia de que se ha realizado dicha devolución.

En lo referente a la creación de clientes, existen dos formas de crearlos:

- Automática: Esto ocurre cuando en alguna de las tiendas se da de alta un cliente que necesita que su compra lleve factura asociada. Entonces, en este caso

cuando la tienda termina el día, se envían al programa todas las altas de cliente realizadas y se insertan de manera inminente.

- Manual: En este caso la creación de clientes tiene más posibilidades, es decir, antes de crearlo se debe indicar que tipo cliente es, ya que en este caso existen los siguientes tipos: socio, cliente de mayor, cliente de taller, empleados, clientes Web. Una vez indicado el tipo, se le asigna un número a dicho cliente y ahora ya se puede rellenar la ficha. Entonces, ese cliente puede ir a cualquier tienda a realizar una compra, debido a que la tienda tiene la opción de traerse del programa ese cliente para poder guardarlo en su base de datos y así poder realizar la factura a su nombre. Por otro lado, dependiendo del tipo de cliente se le hará un descuento en los productos que compre.

3.1.3 Compras y Pagos

En este módulo se realizan las siguientes funciones: creación de pedidos de compra, facturar y abonar pedidos de compra, realizar asientos en el diario de cobros para liquidar el pago de la factura de compra y la creación de proveedores.

En lo referente a la creación de pedidos se hace de la siguiente manera:

En primer lugar se debe indicar en la cabecera del pedido a que proveedor corresponde, para poder insertar en la cabecera las condiciones que se tienen negociadas para el pago del pedido. Dichas condiciones son: fecha vencimiento para el pago del pedido, forma de pago, condiciones de pago y dependiendo del tipo de proveedor se indica a que cuenta se va a imputar dicho pago. Todos estos campos son editables, es decir, se puede dar el caso de que un pedido se haya negociado con otras condiciones distintas a las habituales y se deban modificar los campos anteriormente citados.

A continuación, en las líneas del pedido se deben ir insertando los productos que se quieren incluir en dicho pedido. Al insertar el producto, en una serie de campos de las líneas del pedido se rellenan con los datos que tiene en ese momento el producto en la ficha. Los campos a los que se hace referencia son: Coste neto del producto, Precio de oferta y Precio Psicológico. A partir de esos datos y de que se indique el descuento que se le aplica al producto, se calcula el importe que va a costar comprar el producto.

Una vez completado el relleno del pedido y dependiendo del destino de la mercancía se debe hacer lo siguiente:

- Si el pedido es una compra directa a proveedor que se ha llevado directamente a la tienda, lo que se debe hacer es: En primer lugar, dar por sentado que se ha recibido. Este proceso genera un albarán donde se indican los productos y las cantidades que se han cargado en la tienda correspondiente. A continuación, se debe facturar dicho pedido. Entonces, a la hora de facturar el sistema comprueba que queda presupuesto suficiente en las fechas que se ha indicado en que se va a hacer el pago. En lo referente a la comprobación, el sistema accede al presupuesto del mes que se quiere hacer el pago y dentro de ese mes, se debe mirar el presupuesto correspondiente a la familia de los productos del pedido. Una vez accedido a ese dato, se debe comprobar si el presupuesto que queda es suficiente para realizar el pago del pedido. En caso de no ser suficiente, el sistema emite un mensaje diciendo que para ese mes no se puede imputar el pago y se deben cambiar las condiciones de la cabecera del pedido. En caso de ser suficiente, el sistema factura dicho pedido y genera un documento de cartera pendiente de liquidar, que en el momento de hacer el pago de la factura se debe liquidar dicho documento.
- Si el pedido de compra corresponde al almacén central, el proceso a seguir es el siguiente: En primer lugar, el pedido se debe meter en una recepción de almacén e indicar que cantidades han venido según el albaran. Una vez configurada la recepción, el personal de almacén debe recontar la mercancía para certificar que las cantidades que venían en el albarán son correctas. En caso de que exista alguna incidencia, el personal de almacén se debe poner en contacto con el personal de compras y transmitirle la incidencia que puede deberse a varios motivos: que algún producto esté roto, que venga más cantidad de la indicada o que venga menos cantidad de la indicada. Una vez transmitidas las incidencias, el personal de compras debe reconfigurar la recepción para poder registrarlo de manera correcta. Antes de registrar, se deben indicar las cantidades que se deben mandar a cada zona, debido a que existen varias zonas donde mandar la mercancía: Cross- Docking: En este caso, se reparte una serie de cantidades a las tiendas debido a que son novedades y se quiere que esté en las tiendas lo antes posible. Picking: En este caso, según se haya configurado en la ficha del producto la cantidad máxima que se quiere tener en la estantería para cuando la tienda haga un pedido se sirva de allí. El sistema calcula cuanta cantidad va

destinada para la estantería. Reserva: En caso que quede cantidad pendiente, se destina la cantidad para montar un palet y tenerlo en reserva, para cuando la cantidad en estantería se agote, se baje dicho palet y se rellene la estantería. Una vez administradas las cantidades, se procede a registrar la recepción para realizar el proceso de recibir, el cual genera un albarán donde indica las cantidades recibidas. A continuación, se debe facturar dicho pedido. Entonces, a la hora de facturar el sistema comprueba que queda presupuesto suficiente en las fechas que se han indicado en que se va a hacer el pago. En lo referente a la comprobación, el sistema accede al presupuesto del mes que se quiere hacer el pago y dentro de ese mes, se debe mirar el presupuesto correspondiente a la familia de los productos del pedido. Una vez accedido a ese dato, se debe comprobar si el presupuesto que queda es suficiente para realizar el pago del pedido. En caso de no ser suficiente, el sistema emite un mensaje diciendo que para ese mes no se puede imputar el pago y se deben cambiar las condiciones de la cabecera del pedido. En caso de ser suficiente, el sistema factura dicho pedido y genera un documento de cartera pendiente de liquidar, que en el momento de hacer el pago de la factura se debe liquidar dicho documento.

En lo referente a la facturación de los pedidos se realiza de la siguiente manera:

Una vez que el pedido se haya recibido, se debe avisar al responsable de compras que ya puede facturar dicho pedido. Lo puede hacer de dos maneras: crea una nueva factura y en ella indica que traiga todas las líneas pendientes de facturar de dicho proveedor. Entonces el sistema indica en la factura a que pedido corresponde las líneas posteriores, para que pueda testificar que la facturación es correcta. La otra manera de hacer la facturación, es ir al propio pedido e indicar que se quiere facturar. Además, en cualquier factura se debe comprobar que exista presupuesto en el mes correspondiente al cual se quiere imputar dicha factura. En caso de que no haya suficiente presupuesto, emite un mensaje indicando que debe cambiar las condiciones de pago, es decir, cambiar el mes de imputación.

En lo referente a abonar los pedidos se puede hacer de la siguiente manera:

Se genera un pedido de devolución al proveedor y se van introduciendo de forma manual cada una de las líneas que se quieren devolver al proveedor. Una vez acabado se recibe y factura dicho pedido. La otra manera de realizarlos es: en los pedidos de

devolución existe una opción de copiar líneas de un pedido. Entonces el responsable indica a que pedido corresponde y en ese momento se añaden esas líneas al pedido. Las líneas que no se quieran abonar, el responsable puede borrarlas sin problemas. Una vez hecho ese proceso, se recibe y registra el pedido.

En lo referente a los diarios de pagos existen dos tipos de asientos contables referentes a ese diario:

- Automáticos: Estos asientos se generan y registran de manera automática cuando se factura un pedido, se genera una serie de asientos que se van imputando en la cuenta de IVA, en la cuenta del tipo de proveedor que se ha facturado el pedido y la cuenta de tipo de producto que se indicó en el pedido. Por otro lado, a la hora de facturar una factura que se creó de manera manual se generan los mismos asientos que al facturar un pedido.
- Manuales: Estos asientos los genera el personal de contabilidad y su origen puede ser por varios motivos: cuando se ha pagado la factura del pedido al proveedor, se debe generar un asiento en el diario de pagos para dejar constancia del pago de dicha factura. Otro motivo es cuando se ha restado en el banco una remesa de pagarés con los que se pagó a los proveedores, se debe generar un asiento en el cual se certifique dicho movimiento. Por otro lado, cuando se quiere regular la cuenta de un proveedor o un banco, se debe generar un asiento en el diario de pago para que quede constancia de dicha regulación. Además, cuando el proveedor ha abonado una devolución, se debe generar ese asiento contable para dejar constancia de que se ha recibido dicho abono.

En lo referente a la creación de proveedores, se realiza de la siguiente manera:

En este caso la creación de proveedores tiene más posibilidades, es decir, antes de crearlo se debe indicar que tipo proveedor es, ya que en este caso existen los siguientes tipos: acreedor o proveedor. Una vez indicado el tipo, se le asigna un número a dicho proveedor y ahora se puede rellenar la ficha. Entonces, se debe indicar el grupo contable de negocio de producto al que pertenece el proveedor. Además, se deben indicar los términos de pago y las condiciones de pago que se han negociado con el proveedor. Una vez indicados esos campos, el proveedor queda bien configurado.

3.1.4 Almacén

En este módulo se realizan las siguientes funciones: picking, devoluciones, reaprovisionamiento y recepciones.

En lo referente al picking, se realizan las siguientes acciones:

En primer lugar, el picking se refiere a órdenes que se generan para coger producto de la estantería y gestionar los pedidos referentes a las tiendas y poder servir mercancía a las mismas.

Para generar dichas órdenes, el jefe de almacén debe acceder a una pantalla donde indica: tiendas a las que quiere generar órdenes. Dentro de esa tienda, indicar que transferencias se quieren procesar. En este caso, existen distintos tipos de transferencias, que son los siguientes:

- Reparto Manual: En este caso, el responsable de almacén decide hacer un reparto masivo a todas las tiendas de un producto que es novedad o que se está vendiendo bien y quiere que se reparta en grandes cantidades. Al ejecutar ese reparto, lo que se crean son transferencias con destino a cada una de las tiendas que se le ha indicado cantidad en el reparto.
- Manual: En este caso, al responsable de almacén le han pedido con urgencia una serie de productos desde una tienda. Entonces, para agilizar el proceso puede crear una nueva transferencia de modo manual e insertar los productos que quiere servir a la tienda correspondiente.
- Pedido Tienda: En el programa de la tienda, el encargado puede realizar pedidos de productos que tienen mucha demanda y no le queda nada en la tienda. Una vez insertados todos los productos que necesita, envía el pedido a central para que posteriormente sea procesado
- Reposición de Ventas: En este caso, se procesan las ventas que ha tenido la tienda el día anterior y se repone la cantidad vendida a partir de lo siguiente: a partir de la cantidad vendida ese día, el sistema calcula el stock disponible de la tienda y si tiene stock suficiente para vender en tres días la cantidad que vendió el día anterior, ese producto no se repone. En caso contrario, se debe reponer la cantidad vendida. Entonces, los productos que se deben ir reponiendo se van

insertando en una nueva transferencia para que posteriormente se realice la preparación de la misma.

Una vez que el jefe de almacén ha configurado todos esos datos, ahora debe ejecutar la opción de generar que hace lo siguiente:

Se va recorriendo las tiendas que han sido seleccionadas. Entonces, para cada tienda se recorre todas las transferencias pertenecientes a esa tienda.

Para cada línea de las transferencias comprueba si se ha preparado ya el producto o si la línea está metida en un envío. En caso de que cumpla alguno de los dos requisitos, esa línea de transferencia se obvia. En caso contrario, esa línea se incluye en un envío agrupado donde se irán metiendo todas las líneas pendientes de procesar de dicha tienda. Este proceso se realiza para cada una de las tiendas que se han marcado en la lista.

Una vez procesadas todas las tiendas, el jefe de almacén debe ejecutar la opción de procesar que realiza lo siguiente:

Se va recorriendo las tiendas que han sido seleccionadas. Entonces, para cada tienda se recorre todos los envíos pertenecientes a esa tienda.

Para cada línea de los envíos comprueba si se ha preparado ya el producto. En caso de que cumpla el requisito, esa línea de envío se obvia. En caso contrario, el sistema busca si se puede preparar el producto, es decir, el sistema busca si hay cantidad de ese producto en la estantería. En caso de que haya, el sistema comprueba qué cantidad está libre, debido a que se ha podido reservar cantidad para otra tienda anteriormente. Una vez calculada la cantidad libre de ese producto, la siguiente comprobación es si se puede servir toda la cantidad del pedido. En caso de no ser suficiente, se le servirá la cantidad libre que queda del producto.

A continuación, esa reserva de cantidad se inserta en una tabla con un número de documento específico, para luego a la hora de crear las órdenes se identifiquen por ese número. Este proceso se hace con cada una de las líneas de envío que corresponden a dicha tienda. Una vez procesadas todas las tiendas, se deben generar las órdenes de picking para que se puedan preparar los productos. Entonces, el sistema accede a la tabla donde se han hecho las inserciones y filtra por el número de documento correspondiente. A continuación, se va accediendo a cada uno de los registros que se encuentran dentro del filtro y se va generando la orden de picking correspondiente.

Una vez terminado el proceso y habiendo generado las órdenes de picking para todas las tiendas, el jefe de almacén debe asignar esas órdenes a un operario de almacén. Para ello, tiene que ir a la cabecera de las órdenes e indicar a que operario le quiere asignar las órdenes. Una vez esté asignado, el sistema le asigna todas las órdenes correspondientes a esa cabecera.

A continuación, los operarios de almacén para preparar esos productos trabajan con PDA, la cual se conecta al programa mediante un servicio WEB (que se explicará más adelante) que se instala en la PDA. Entonces, el operario entra en la opción picking y señala la tienda que tiene que preparar y el sistema obliga a indicar un número de palet, ya que a cada producto preparado, el sistema lo mete en ese palet. Una vez acabada la preparación, el operario va al ordenador, busca el palet y hay una función que es cerrar bulto, que lo que hace es descontar esos productos del stock de almacén y cargarlos en el stock de la tienda.

En lo referente a las devoluciones, se realizan de la siguiente manera:

Existen dos tipos de devoluciones, que cada una se realiza de una manera:

- Devoluciones Tiendas: En las tiendas tienen habilitada una opción donde pueden realizar devolución de los productos. En primer lugar, se debe crear una nueva transferencia de devolución. Una vez creada, el encargado de la tienda debe insertar los productos que quiere devolver. En este caso, lo puede hacer manualmente tecleando el código interno del producto o mediante el lector de código de barras pasan el producto y el programa busca a que código interno está asociado. Una vez insertado el código, se debe decir la cantidad a devolver. A continuación, se debe indicar que destino tendrá la cantidad ya que existen varios destinos:
  - Rotura: En este caso el producto ha venido roto y se debe indicar que el destino es roturas, para luego crear un documento devolución al proveedor por rotura.
  - Robo: En caso de que se haya producido un robo y el encargado sepa que productos han sido robados, se debe indicar que el producto se debe cargar en el almacén robos, para su posterior tratamiento en el almacén Central.
  - Otra Tienda: En este caso, una tienda necesita un producto urgente para venderlo a un cliente. Entonces, la tienda se encarga de contactar con otras y

pedirles el producto que quiere. En caso de que una tienda lo tenga, la tienda debe abrir una transferencia e indicar a que tienda va dirigido el producto.

- Exceso: En este caso, si un producto se encuentra fuera de temporada se debe devolver a central. Entonces, el encargado debe indicar que el producto se devuelve por exceso a central. Por otro lado, a la hora de registrar la transferencia le pedirá un código de autorizador, debido a que el jefe de compras se quiere asegurar que devuelven los productos correctos.
- Proveedor: En este caso, si un producto no ha tenido éxito y el jefe de compras ha negociado con el proveedor la devolución del mismo, entonces, el encargado debe indicar que ese producto es una devolución al proveedor, para que el personal del almacén central gestione dicha devolución.
- Taller: En este caso, si el producto corresponde a una reparación que debe hacer taller, el encargado de tienda debe indicar que el destino es taller. Así, cuando llegue a central el responsable de devoluciones sepa que ese producto debe entregarlo en taller.

Una vez insertados todos los productos en la transferencia, se debe registrar dicha transferencia para que se cargue en central. Entonces, una vez que la mercancía ha llegado a central, el encargado de devoluciones debe hacer lo siguiente:

Debe buscar en el módulo devoluciones la transferencia que le ha llegado. Una vez encontrada, debe habilitar el conteo para poder certificar que lo que registró en la transferencia el encargado es correcto. En caso de ser correcto, se registra la transferencia y dependiendo del destino realiza una cosa u otra. A continuación se explicará que se hace para cada caso:

- Roturas: En este caso, se resta de la tienda la cantidad indicada en la transferencia y se carga en el almacén roturas.
- Robos: En este caso, se resta de la tienda la cantidad indicada en la transferencia y se carga en el almacén robos.
- Otra tienda: En este caso, se resta de la tienda origen la cantidad indicada en la transferencia y se carga en la tienda destino.
- Exceso: En este caso, se resta de la tienda la cantidad indicada en la transferencia y se carga en el almacén Central. Además, se crea un documento de ubicación para que el operario de almacén, mediante la PDA, coloque el producto en la estantería correspondiente.

En caso de haber incidencias en el conteo de la mercancía, se debe hacer lo siguiente:

En caso de que el recuento sea menor que lo que decía de inicio, el restante se debe volver a cargar en la tienda para que no haya descuadre de stock debido a un error humano. Por otro lado, si el conteo es menor debido a que en algunos productos su código interno era distinto al que se había marcado, entonces, lo que se debe hacer es añadir a la transferencia esos códigos para que luego al registrar quede cuadrado el stock de la tienda.

En caso de que el recuento sea mayor, el sistema creará una nueva línea donde se deberá indicar la cantidad que ha venido de más de ese producto. Por otro lado, el responsable debe indicar el destino que tendrá la nueva línea, que por lógica será el mismo que la línea original.

Una vez resueltas las incidencias, se debe proceder al registro de la transferencia para que la mercancía pueda ser manipulada y gestionada según el destino que tenga dicha mercancía.

- Devolución de clientes: En este caso, lo primero que debe hacer el responsable de devoluciones es insertar los productos en un nuevo abono de cliente. Una vez insertados, avisa al responsable de ventas a clientes para que configure cómo se quiere registrar el abono. El único dato que no se debe modificar es que esa mercancía se debe cargar en el almacén franquicia

Una vez registrado el abono, el encargado de devoluciones debe crear una nueva transferencia con origen franquicia y destino devoluciones. A continuación, se debe empezar a pasar la mercancía para que el sistema vaya insertando las líneas de los productos o aumentando la cantidad si ya existe el producto creado. Entonces, una vez cotejada la mercancía se debe decidir cual debe ser el destino de la mercancía, como se ha explicado anteriormente en las devoluciones a tiendas. Por último, una vez configuradas las líneas de la transferencia, se debe registrar dicha transferencia para poder manipular la mercancía.

En lo referente al reaprovisionamiento, se realiza de la siguiente manera:

Una vez que el jefe de almacén ha creado los envíos donde se agrupa todo lo que se debe preparar a la tienda y ha creado las órdenes para realizar la preparación de la

tiendas, tiene la posibilidad de reponer los productos de los que no hay suficiente stock para satisfacer la demanda que tienen las tiendas. El proceso es el siguiente:

El sistema recorre todas las líneas de envío y por cada producto inserta un registro en una tabla temporal, donde se guarda la siguiente información: cantidad de envío, cantidad en picking, cantidad necesaria y cantidad servida. Entonces, la primera vez que se encuentra el producto se calcula la cantidad en picking y se inserta el registro con cantidad en picking calculada, con cantidad en envío que tenga esa línea y se calcula cantidad necesaria haciendo la resta de cantidad en envío y cantidad en picking. Para posteriores búsquedas, únicamente se actualiza la cantidad en envío y la cantidad necesaria. Este proceso se hace para cada uno de los productos que se encuentran en las líneas de envío.

A continuación, el sistema se recorre la tabla temporal donde están insertados los productos y comprueba si necesita que se reaprovisione la estantería porque no hay cantidad suficiente para satisfacer la cantidad que existe en pedidos de las diferentes tiendas. En caso de necesitar reaprovisionamiento el producto, el sistema recorre los palets de reserva que contienen ese producto y comprueba que el producto se encuentra en reserva. En caso de encontrar alguno, se crea un movimiento para bajar el bulto de la reserva y así poder reaprovisionar la estantería. En una variable se guarda la cantidad localizada y una vez generado el movimiento, se comprueba si se necesita de otro palet para satisfacer la cantidad necesaria para reaprovisionar. En caso de necesitarlo, se realiza el proceso anteriormente explicado. Todo este proceso se realiza para cada uno de los registros de la tabla

A continuación, un operario montando en la trilateral entra en la PDA donde va indicando los palets a bajar y de que ubicación debe cogerlos. Entonces, el operario se dirige a la ubicación y comprueba, físicamente, que el palet se encuentra ahí. Entonces, en la PDA confirma el palet. Entonces, el sistema genera un movimiento que quita el producto de esa ubicación de reserva y lo mete en una ubicación ficticia que se llama isla. El operario debe hacer este proceso con todos los bultos que vaya indicando el sistema.

Una vez manipulados todos los palets que tenían creado un movimiento, otro operario va con la PDA y marca el palet. Entonces, el sistema marca el pasillo al que debe ir el producto y la cantidad que existe en el palet. A continuación, el operario debe dirigirse al pasillo que ha marcado el sistema y buscar una ubicación adecuada donde

colocar el producto. Una vez buscada la ubicación, el operario debe marcar en la PDA la ubicación donde lo coloca y la cantidad que va a ubicar. Entonces, el sistema da de baja el producto de la zona ficticia donde estaba y lo da de alta en la ubicación. Si la cantidad que ha indicado es el total que había en el palet el reaprovisionamiento ha terminado. En caso de que no fuera la totalidad, el sistema marca la cantidad que queda en el palet. Entonces, el operario debe decidir que hacer: si ubicar el restante o subir el palet a la reserva. En caso de querer subirlo, debe entrar en otra opción donde el sistema busca ubicación al palet para que el operario de la trilateral lo ubique después. En caso de querer ubicar el resto, se debe seguir el proceso explicado anteriormente. Por otro lado, si se da el caso de que la cantidad física difiere de lo que dice el programa, el operario debe ubicar todas las unidades físicas y las restantes que indicar la PDA se deben bloquear para que el responsable regularice esas unidades. Este proceso debe realizarlo para cada uno de los palets con orden de reaprovisionamiento.

En lo referente a las recepciones, se realizan de la siguiente manera:

En primer lugar, los operarios deben descargar el camión o el trailer en que ha llegado la mercancía. Una vez descargado, el transportista entrega el albarán donde indica a que pedido corresponde ese material. Entonces, dicho albarán se debe entregar al personal de compras para que con el número de pedido, cree una nueva recepción donde estarán los productos a recontar. Además, una vez se haya creado la recepción se debe configurar que reparto se quiere hacer, es decir, indicar cuanta mercancía se servirá directamente a las tiendas. A continuación, los empleados de almacén deben recontar los productos para corroborar que las cantidades que vienen indicadas son las correctas o existe alguna incidencia. En caso de no existir ninguna incidencia, el personal de compras debe registrar la recepción. Entonces, al registrar el sistema hace lo siguiente para cada producto:

En primer lugar carga la cantidad en el almacén central. A continuación, carga la cantidad en la zona de recepción. Una vez cargado, se crea un documento de ubicación para indicar dónde debe ir la mercancía. En este caso, lo primero que se ejecuta es el reparto y se van creando líneas en el documento de ubicación indicando que se va a sacar de la zona de recepción la cantidad repartida para dicha tienda y se va a cargar en la zona de expedición en la ubicación correspondiente a la tienda. Una vez generadas las líneas para cada tienda con la cantidad restante se hace lo siguiente:

En la ficha de producto mira la cantidad máxima que se quiere tener en picking de ese producto. Además, calcula la cantidad que hay en estos momentos en picking. Por lo tanto, hace la resta entre la cantidad máxima y lo que hay en picking en esos momentos. Si la resta es mayor que 0 se crean en el documento de ubicación las líneas donde se indicará la cantidad destinada a picking. En caso de que siga faltando cantidad por asignar, todo ese restante se mandará a reserva.

Una vez que se haya hecho todo este proceso para cada uno de los productos, ya se puede manipular la mercancía para colocar en su destino correspondiente. Entonces, dependiendo del destino se hace lo siguiente:

Directo a Tienda: En este caso, toda la cantidad que se va a servir a tienda, se lleva a una zona especial para que pueda ser manipulada. Por lo tanto, el operario de esa zona coge el producto y en una pantalla del programa pasa el producto por el lector de código de barras. Entonces, al pasarlo el sistema le muestra las líneas correspondientes al envío directo a tienda y le indica la cantidad total. Además, el operario que previamente contó la cantidad que tenía, debe marcar dicha cantidad y darle a registrar. Dicho registro hace lo siguiente:

Se va recorriendo cada una de las líneas correspondientes a ese producto destinadas al envío directo y en la columna cantidad a enviar lo rellena con la cantidad destinada a dicha tienda siempre y cuando la cantidad pendiente de manipular sea mayor que la cantidad destinada. En caso de no ser mayor, se rellena con la cantidad pendiente de manipular. A continuación, registra dicha línea y lo que hace es quitar de la zona de recepción la cantidad a enviar que se ha indicado y la carga en la zona de expedición. Después, crea un palet virtual que corresponde a la tienda e inserta dicho producto. Este palet se crea para que luego al final de la jornada se cierre y se cargue el stock a la tienda y se pueda tener control de lo que se va mandando a las tiendas. Una vez insertado el producto en el palet, se imprimen tantas pegatinas como cantidad a enviar se indicó para que luego se coloque en la ubicación que corresponda. Este proceso se hace hasta que la cantidad pendiente de manipular sea cero.

Picking: En este caso, un operario de la zona de Picking entra en la PDA y lee el producto. La PDA le indica la cantidad que debe colocar y a que pasillo debe ir para ubicar dicha mercancía. Entonces, el operario va al pasillo indicado y busca una ubicación donde colocar la mercancía. Cuando ya encontró la ubicación adecuada, debe marcar el producto en la PDA, marcar la ubicación y la cantidad que se quiere ubicar.

Una vez marcados esos datos, debe clicar en el botón aceptar. Entonces, el sistema hace lo siguiente:

Quita de la zona de recepción la cantidad marcada y la carga en la zona de picking en la ubicación que señaló.

Reserva: En este caso, un operario de la zona de recepción entra en la PDA y lee el producto. La PDA le indica la cantidad destinada a reserva. A continuación, el operario debe marcar el palet donde va a colocar la mercancía. Después, debe indicar cuanta cantidad va destinada a ese palet y dar al botón aceptar. Entonces, el sistema resta del total destinado a reserva la cantidad que se ha indicado anteriormente. En caso de quedar mercancía pendiente, se debe realizar el proceso anteriormente citado. Por otro lado, a los palets que se van generando se les debe buscar ubicación para que el operario de la trilateral los coloque en la ubicación adecuada y registrar ese movimiento. Al registrar ese movimiento provoca lo siguiente: Quita de la zona de recepción la cantidad marcada y la carga en la zona de reserva en la ubicación que señaló.

3.1.5 Tienda

En este apartado se realizan las siguientes funciones: apertura de caja, venta, abono, reserva, pedidos, transferencias.

En lo referente a las aperturas, se realiza de la siguiente manera:

Cada día, el encargado de la tienda realiza la apertura del día informáticamente, para que luego a la hora de realizar ventas, abonos y reservas el sistema sepa a que fecha debe imputar dichas operaciones. En caso de no realizar la apertura del día, el sistema no deja realizar ninguna de las operaciones citadas anteriormente. En primer lugar, el sistema pregunta al usuario si quiere sincronizar los productos que actualizaron las personas de compras. El usuario debe sincronizar para poder tener los precios correctamente.

En lo referente a las ventas, se realiza de la siguiente manera:

En la pantalla de ventas se muestra lo siguiente:

Cabecera Venta: En esta parte se muestran los datos del cliente al que se le va hacer la venta. Por defecto, el cliente es un cliente genérico donde se imputan todas las ventas que va a hacer la tienda. Si un cliente quiere factura de venta, se da la posibilidad de cambiar ese cliente genérico por el cliente real en la cabecera.

Líneas Ventas: En este parte el personal de caja debe ir insertando los productos que se van a vender al cliente. Para identificar los productos se puede hacer de varias maneras:

Código de barras: En este caso, el vendedor pasa el código de barras del producto por el lector de códigos. Entonces, el sistema comprueba en una tabla si existe ese código de barras en la base de datos de la tienda. En caso de existir, el sistema accede a la ficha del producto para obtener el precio del mismo. Una vez obtenido el precio, se inserta el producto en la línea de venta. En caso de no encontrar el código de barras, se debe insertar otro dato del producto para poder encontrarlo.

Referencia del Proveedor: En este caso, el vendedor introduce la referencia del proveedor del producto. Entonces, el sistema busca todos los productos con esa referencia que se ha introducido. Se pueden dar dos casos: Que exista un único producto con esa referencia y entonces lo inserta en la línea de venta o existen varios productos con la misma referencia. Por lo tanto, el sistema muestra los productos con esa referencia y el vendedor debe elegir uno de ellos. Una vez elegido, el sistema lo inserta en la línea de venta.

Código Interno: En caso de que el producto no se haya encontrado ni por referencia ni por código de barras, el vendedor puede buscar el producto por descripción y si lo encuentra, debe señalar a qué código interno del producto se refiere para que el sistema lo inserte en la línea de venta.

Una vez identificado el producto, se debe indicar la cantidad que se quiere llevar el cliente.

Este proceso se debe realizar para cada uno de los productos que se quiere llevar el cliente.

Una vez insertados todos los productos del cliente, el vendedor debe pulsar F11 para poder terminar la venta. Entonces, el sistema una vez pulsa la tecla, se recorre todas las líneas de ventas para ir acumulando el importe, para que cuando el programa pase a la siguiente pantalla, el vendedor pueda ver el coste total de la venta.

A continuación, al vendedor se le muestra una pantalla donde puede ver el importe total de la venta, donde debe indicar el número de vendedor y la forma de pago con la que va a pagar el cliente.

En referencia al número del vendedor, se debe hacer referencia ya que el personal de RRHH al final del mes saca un informe donde indica las ventas de cada uno

de los vendedores de la tienda, para luego poder calcular las comisiones que se lleva cada uno.

En cuanto a la forma de pago, existen varias formas:

- Caja: El cliente va a pagar en efectivo.
- Tarjeta: El cliente va a pagar con tarjeta
- BONO: En este caso, cuando se le hace una devolución al cliente, por defecto se le devuelve un BONO que puede canjear en próximas ventas.
- VALE-CC: Es análogo al BONO, pero en este caso corresponde alguna promoción que haya hecho el centro comercial.

En lo referente a las formas de pago, en una venta puede haber distintas formas de pago siempre y cuando esas formas de pago completen el importe total.

Una vez insertados todos esos datos, el vendedor debe pulsar F11 para facturar la venta. En este caso, el sistema genera un movimiento contable de dicha venta, con fecha registro correspondiente al día de la última caja que se ha hecho apertura. Por otro lado, por cada producto genera un movimiento de tipo venta para quitar del stock de la tienda las unidades que acaba de vender. Una vez hechos esos procesos, el sistema manda una orden a la impresora para que imprima el ticket de venta que se entrega al cliente. En caso de que una de las formas de pago sea tarjeta, también se imprime el justificante de pago que firma el cliente.

En lo referente a los abonos, se realiza de la siguiente manera:

En la pantalla de abonos se muestra lo siguiente:

Cabecera Abono: En esta parte se muestran los datos del cliente al que se le va hacer el abono. Por defecto, el cliente es un cliente genérico donde se imputan todos los abonos que va a hacer la tienda. Si un cliente quiere factura del abono, se da la posibilidad de cambiar ese cliente genérico por el cliente real en la cabecera.

Líneas Ventas: En este parte el personal de caja debe ir insertando los productos que se van a abonar al cliente. Para identificar los productos se puede hacer de varias maneras:

Código de barras: En este caso, el vendedor pasa el código de barras del producto por el lector de códigos. Entonces, el sistema comprueba en una tabla si existe ese código de barras en la base de datos de la tienda. En caso de existir, el sistema accede a la ficha del producto para obtener el precio del mismo. Una vez obtenido el

precio, se inserta el producto en la línea de abono. En caso de no encontrar el código de barras, se debe insertar otro dato del producto para poder encontrarlo.

Referencia del Proveedor: En este caso, el vendedor introduce la referencia del proveedor del producto. Entonces, el sistema busca todos los productos con esa referencia que se ha introducido. Se pueden dar dos casos: Que exista un único producto con esa referencia y entonces lo inserta en la línea del abono o existen varios productos con la misma referencia. Entonces, el sistema muestra los productos con esa referencia y el vendedor debe elegir uno de ellos. Una vez elegido, el sistema lo inserta en la línea de abono.

Código Interno: En caso de que el producto no se haya encontrado ni por referencia ni por código de barras, el vendedor puede buscar el producto por descripción y si lo encuentra, debe señalar a qué código interno del producto se refiere para que el sistema lo inserte en la línea del abono.

Una vez identificado el producto, se debe indicar la cantidad que quiere devolver el cliente

Este proceso se debe realizar para cada uno de los productos que quiere devolver el cliente.

Una vez insertados todos los productos del cliente, el vendedor debe pulsar F11 para poder terminar el abono.

A continuación, al vendedor se le muestra una pantalla donde debe indicar el número de vendedor y la forma de pago con la que va a abonar el cliente.

En referencia al número del vendedor, se debe hacer referencia ya que el personal de RRHH al final del mes saca un informe donde indica las ventas de cada uno de los vendedores de la tienda, para luego poder calcular las comisiones que se lleva cada uno.

En cuanto a la forma de pago, existen varias formas:

- Caja: Se le va a abonar al cliente en efectivo.
- Tarjeta: Se le va a abonar al cliente en tarjeta.
- BONO: En este caso, cuando se le hace una devolución al cliente, por defecto se le devuelve un BONO que puede canjear en próximas ventas.

Una vez insertados todos esos datos, el vendedor debe pulsar F11 para registrar el abono. En este caso, el sistema genera un movimiento contable del abono, con fecha registro correspondiente al día de la última caja que se ha hecho apertura. Por otro lado,

por cada producto genera un movimiento de tipo abono para cargar en el stock de la tienda las unidades que acaba de abonar. Una vez hechos esos procesos, el sistema manda una orden a la impresora para que imprima el abono que se le debe entregar al cliente.

Otra forma de realizar el abono es a partir de una factura. En la cabecera del abono existe un campo donde se puede insertar el número de ticket de venta, por la cual se le quiere hacer el abono. Una vez insertado, el sistema inserta en las líneas del abono cada uno de los productos que corresponden a la factura. Entonces, puede haber dos casos: que se quiere abonar la totalidad de los productos o solo se quiere abonar una parte de los productos. En caso de que se quiera abonar una parte, el vendedor debe borrar aquellos productos que no van a ser abonados. Una vez realizado eso, el proceso es análogo a lo explicado anteriormente.

En lo referente a las reservas se hace de la siguiente forma:

Una reserva se crea si un cliente viene buscando un producto y en la tienda no se encuentra pero el vendedor sabe que pronto lo va a recibir y al cliente le urge tenerlo. Entonces, se crea una reserva y los pasos a seguir para crearla son los siguientes:

Cabecera Reserva: En esta parte se muestran los datos del cliente al que se le va a hacer la reserva. Además, se debe indicar un cliente específico, ya que si posteriormente viene a recoger su pedido, se puede identificar fácilmente la reserva. Por otro lado, se debe indicar la forma de pago de la señal que va a dejar el cliente.

En cuanto a la forma de pago, existen varias formas:

- Caja: El cliente va a pagar en efectivo.
- Tarjeta: El cliente va a pagar con tarjeta
- BONO: En este caso, cuando se le hace una devolución al cliente, por defecto se le devuelve un BONO que puede canjear en próximas reservas.

Además, se debe indicar el número de vendedor que va a realizar la reserva.

Líneas Reserva: En este parte el personal de caja debe ir insertando los productos que se van a reservar al cliente. Para identificar los productos se puede hacer de varias maneras:

Código de barras: En este caso, el vendedor pasa el código de barras del producto por el lector de códigos. Entonces, el sistema comprueba en una tabla si existe ese código de barras en la base de datos de la tienda. En caso de existir, el sistema

accede a la ficha del producto para obtener el precio del mismo. Una vez obtenido el precio, se inserta el producto en la línea de la reserva. En caso de no encontrar el código de barras, se debe insertar otro dato del producto para poder encontrarlo.

Referencia del Proveedor: En este caso, el vendedor introduce la referencia del proveedor del producto. Entonces, el sistema busca todos los productos con esa referencia que se ha introducido. Se pueden dar dos casos: Que exista un único producto con esa referencia y entonces lo inserta en la línea de la reserva o existan varios productos con la misma referencia. Entonces, el sistema muestra los productos con esa referencia y el vendedor debe elegir uno de ellos. Una vez elegido, el sistema lo inserta en la línea de la reserva.

Código Interno: En caso que el producto no se haya encontrado ni por referencia ni por código de barras, el vendedor puede buscar el producto por descripción y si lo encuentra, debe señalar a qué código interno del producto se refiere para que el sistema lo inserte en la línea de la reserva.

Una vez identificado el producto, se debe indicar la cantidad del mismo que se quiere reservar al cliente.

Este proceso se debe realizar para cada uno de los productos que se quiere llevar el cliente.

Una vez insertados todos los productos del cliente, el vendedor debe pulsar F11 para poder registrar la reserva.

En este caso, el sistema genera un movimiento contable de dicha reserva, con fecha registro correspondiente al día de la última caja en que se ha hecho apertura. Una vez hechos esos procesos, el sistema manda una orden a la impresora para que imprima el ticket de reserva que se le debe entregar al cliente. Además, se imprime una copia para que la tienda se lo quede y lleva un control de las reservas pendientes que tiene aún por procesar.

Por otro lado, existe la opción de aumentar la señal que dejó de primeras el cliente. En el formulario de reservas existe una opción que es aumentar reserva. En primer lugar, el vendedor debe buscar la reserva correspondiente. Una vez buscada, se señala la opción de aumentar reserva. Entonces, en la cabecera debe indicar en que forma de pago hará dicho aumento y el importe que quiere dejar. Una vez indicado eso, se registra el aumento. Antes de realizar algún movimiento contable, el sistema comprueba que dicho aumento unido a la anterior señal no sume el importe total. En

caso de que sea así, muestra un mensaje de error indicando que no puede darse ese caso. Una vez hecha la comprobación, el sistema registra el aumento de dicha reserva.

Por último, existe la opción de convertir a ticket la reserva, es decir, el caso que el producto de la reserva haya llegado a la tienda, el vendedor avisa el cliente para que venga a recoger su pedido. El sistema lo que hace es convertir esa reserva como si fuera una venta normal y corriente, lo único que cambia es que a la hora de mostrar el importe total que se debe facturar al cliente, se resta el importe acumulado de las señales y muestra lo que resta por pagar al cliente. Quitando esa salvedad, el proceso de facturar esa venta es análogo al explicado anteriormente.

En lo referente a pedidos, se realiza de la siguiente manera:

En el programa existe una opción que es pedidos de tienda. Al pulsar sobre esa opción, el sistema muestra los pedidos de tienda que no han sido registrados por el encargado de la tienda. En primer lugar, el encargado debe seleccionar la opción de nuevo pedido. Entonces, el sistema crea una nueva cabecera de pedido y se lo muestra al encargado. Una vez se haya creado, el encargado puede ir insertando los productos que necesita tener en la tienda, que no tiene o están bajo mínimos. Para insertarlos, debe hacerlo de la siguiente manera:

Código de barras: En este caso, el encargado pasa el código de barras del producto por el lector de códigos. Entonces, el sistema comprueba en una tabla si existe ese código de barras en la base de datos de la tienda. Una vez identificado, se inserta el producto en la línea del pedido. En caso de no encontrar el código de barras, se debe insertar otro dato del producto para poder encontrarlo.

Referencia del Proveedor: En este caso, el encargado introduce la referencia del proveedor del producto. Entonces, el sistema busca todos los productos con esa referencia que se ha introducido. Se pueden dar dos casos: Que exista un único producto con esa referencia y entonces lo inserta en la línea del pedido o existan varios productos con la misma referencia. Entonces, el sistema muestra los productos con esa referencia y el encargado debe elegir uno de ellos. Una vez elegido, el sistema lo inserta en la línea del pedido.

Código Interno: En caso de que el producto no se haya encontrado ni por referencia ni por código de barras, el encargado puede buscar el producto por

descripción y si lo encuentra, debe señalar a que código interno del producto se refiere para que el sistema lo inserte en la línea de pedido.

Una vez identificado el producto, se debe indicar la cantidad que se quiere pedir al almacén.

Una vez insertados todos los productos, se debe registrar el pedido. El sistema lo que hace en este caso y mediante un conector, es mandar el pedido a central para que pueda ser procesado por el almacén.

En lo referente a las transferencias, se realiza de la siguiente manera:

En el programa existe una opción que es transferencia a central. Al pulsar sobre esa opción, el sistema muestra las transferencias a central que no han sido registradas por el encargado de la tienda. En primer lugar, el encargado debe seleccionar la opción de nueva transferencia. Entonces, el sistema crea una nueva cabecera de transferencia y se lo muestra al encargado. Una vez se haya creado, el encargado puede ir insertando los productos que van a devolver a central. Para insertarlos, debe hacerlo de la siguiente manera:

Código de barras: En este caso, el encargado pasa el código de barras del producto por el lector de códigos. Entonces, el sistema comprueba en una tabla si existe ese código de barras en la base de datos de la tienda. Una vez identificado, se inserta el producto en la línea de la transferencia. En caso de no encontrar el código de barras, se debe insertar otro dato del producto para poder encontrarlo.

Referencia del Proveedor: En este caso, el encargado introduce la referencia del proveedor del producto. Entonces, el sistema busca todos los productos con esa referencia que se ha introducido. Se pueden dar dos casos: Que exista un único producto con esa referencia y entonces lo inserta en la línea de la transferencia o existan varios productos con la misma referencia. Entonces, el sistema muestra los productos con esa referencia y el encargado debe elegir uno de ellos. Una vez elegido, el sistema lo inserta en la línea de la transferencia.

Código Interno: En caso de que el producto no se haya encontrado ni por referencia ni por código de barras, el encargado puede buscar el producto por descripción y si lo encuentra, debe señalar a que código interno del producto se refiere para que el sistema lo inserte en la línea de la transferencia.

Una vez identificado el producto, se debe indicar la cantidad que se quiere mandar al almacén.

Además, debe indicar el destino de la mercancía, que puede ser el siguiente:

- Rotura: En este caso, el producto ha venido roto y se indica que se debe cargar en el almacén roturas, para luego crear un documento devolución al proveedor por rotura.
- Otra Tienda: En este caso, una tienda necesita un producto urgente para venderlo a un cliente. Entonces, la tienda se encarga de contactar con otras y pedirles el producto que quiere. En caso de que una tienda lo tenga, la tienda debe abrir una transferencia para indicar que el destino es otra tienda e indicar a que tienda va dirigido el producto.
- Exceso: En este caso, un producto que está fuera de temporada y se debe devolver a central. Entonces, el encargado indica que el producto se devuelve por exceso a central. Por otro lado, a la hora de registrar la transferencia le pedirá un código de autorizador, debido a que el jefe de compras se quiere asegurar que devuelven los productos correctos.
- Proveedor: En este caso, si un producto no ha tenido éxito y el jefe de compras ha negociado con el proveedor la devolución del mismo. Entonces, el encargado indica que ese producto es una devolución al proveedor, para que el personal del almacén gestione la devolución.
- Taller: En este caso, si el producto corresponde a una reparación que debe hacer taller, el encargado de tienda indica que el destino es taller. Así cuando llegue a central, el responsable sepa que ese producto debe entregarlo en taller.

Una vez insertados todos los productos, se registra la transferencia. El sistema lo que hace en este caso y mediante un conector, es mandar la transferencia a central para que pueda ser procesada por el almacén

Otro tipo de transferencias son: Transferencia Coordinador, Recibir de Coordinador, Robos. Su funcionamiento es análogo al de las transferencias salvo que no deben indicar el destino de la mercancía, ya que de antemano se sabe el movimiento de stock que se debe hacer.

4 Estructura del Trabajo Realizado

En este apartado se muestran, de manera gráfica, los distintos módulos de que está compuesto el programa y dentro de cada uno de ellos, se muestran las entradas y salidas que se producen y los documentos que se intercambian entre módulos. Por otro lado se explica, de manera más detallada, la funcionalidad de cada módulo.

4.1 Arquitectura de la aplicación

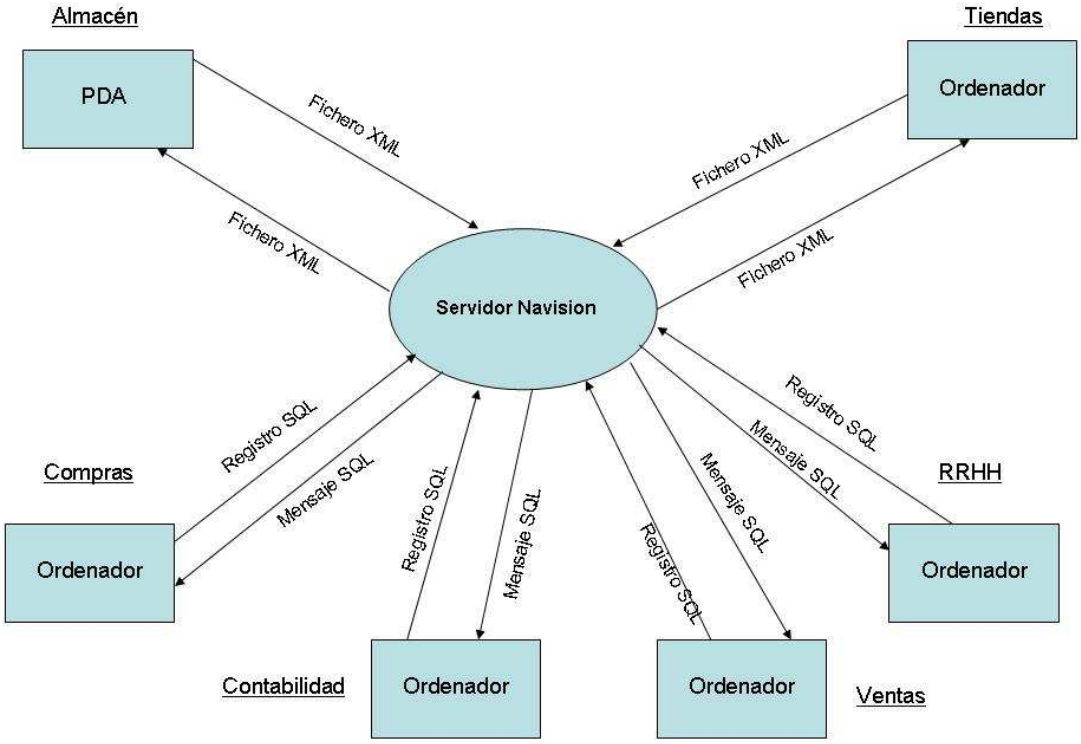


Ilustración 26 Arquitectura de Aplicación

MÓDULOS

Almacén

Entradas: En este caso, para cualquier función que se quiera hacer con la PDA, ya sea información de un producto, información de un palet, registro de movimientos, actualización de datos, etc.., lo que se hace es mandar un fichero XML, mediante un conector, con una serie de parámetros al servidor de Navision para que el mismo pueda procesar dicha información.

Salidas: En este caso, una vez que el servidor de Navision recibe el fichero XML de la PDA, procesa dicha información para hacer las pertinentes consultas en las tablas de Navision. Una vez hechas las consultas, el servidor de Navision le envía la respuesta a la PDA por medio de un fichero XML, el cual puede ser de dos tipos: que contenga error porque la información pedida no existe o el registro no se ha podido hacer o que contenga los datos pedidos en la consulta en caso de ser consulta o que muestra un mensaje de que el registro se ha hecho de forma correcta en el caso de ser registro.

Intercambio de ficheros: Ficheros XML

Tiendas

Entradas: En este caso, para cualquier función que se quiera hacer desde el cliente de Navision, ya sea información del stock en las diferentes tiendas de un determinado producto, actualización de un producto, apertura de la caja del día, registro de venta, registro de abono, registro de reservas, envío de pedidos y transferencias, lo que se hace es mandar un fichero XML, mediante un conector, con una serie de parámetros al servidor de Navision para que el mismo pueda procesar dicha información.

Salidas: En este caso, una vez que el servidor de Navision recibe el fichero XML del ordenador, procesa dicha información para poder hacer las pertinentes consultas en las tablas de Navision. Una vez hechas las consultas, el servidor de Navision le envía la respuesta al ordenador por medio de un fichero XML, el cual puede ser de dos tipos: que contenga error porque la información pedida no existe o el registro no se ha podido hacer o que contenga los datos pedidos en la consulta en caso de ser consulta o que

muestra un mensaje de que el registro se ha hecho de forma correcta en el caso de ser registro.

Intercambio de ficheros: Ficheros XML

Compras

Entradas: En este caso, para cualquier función que se quiere hacer desde el cliente de Navision, ya sea creación de un pedido de compra, inserta una línea en un pedido de compra, crear una recepción de mercancía, registro de un pedido, registro de una factura, lo que se hace es enviar al servidor un registro de SQL para que pueda insertarlo en la tabla correspondiente.

Salidas: En este caso, una vez que el servidor de Navision recibe el registro SQL del ordenador, debe procesar la información para poder hacer las pertinentes inserciones en las tablas de SQL. A continuación el servidor emite un mensaje al cliente, que puede ser para informar que la función se ha hecho correctamente o que se ha producido un error a la hora de hacerlo.

Intercambio de ficheros: En este caso, si en alguna función existe la opción de importar los datos a Excel, el servidor le devuelve una hoja de Excel rellena con los datos.

Ventas

Entradas: En este caso, para cualquier función que se quiere hacer desde el cliente de Navision, ya sea creación de un pedido de venta, inserta una línea en un pedido de venta, crear un envío para que el almacén prepare el pedido, registro de un pedido, registro de una factura. Lo que se hace es enviar al servidor un registro de SQL para que pueda insertarlo en la tabla correspondiente.

Salidas: En este caso, una vez que el servidor de Navision recibe el registro SQL del ordenador, debe procesar dicha información para poder hacer las pertinentes inserciones en las tablas de SQL. A continuación el servidor emite un mensaje al cliente, que puede ser para informar que la función se ha hecho correctamente o que se ha producido un error a la hora de hacerlo.

Intercambio de ficheros: En este caso, si en alguna función existe la opción de importar los datos a Excel, el servidor le devuelve una hoja de Excel rellena con los datos.

Contabilidad

Entradas: En este caso, para cualquier función que se quiere hacer desde el cliente de Navision, ya sea registro de un pago, registro de un cobro, creación de una factura, creación de un diario de pagos, creación de un diario de cobros, modificación de presupuestos, registro de pagarés, creación de norma. Lo que se hace es enviar al servidor un registro de SQL para que pueda insertarlo en la tabla correspondiente.

Salidas: En este caso, una vez que el servidor de Navision recibe el registro SQL del ordenador, debe procesar la información para poder hacer las pertinentes inserciones en las tablas de SQL. A continuación el servidor emite un mensaje al cliente, que puede ser para informar que la función se ha hecho correctamente o que se ha producido un error a la hora de hacerlo.

Intercambio de ficheros: En este caso, si en alguna función existe la opción de importar los datos a Excel, el servidor le devuelve una hoja de Excel rellena con los datos. En caso de que la función sea creación de la norma, el servidor le devuelve un fichero plano con la norma que ha pedido.

RRHH

Entradas: En este caso, para cualquier función que se quiere hacer desde el cliente de Navision, ya sea para dar de alta un vendedor, consultar las ventas por vendedor que se ha hecho en una tienda, consultar las ventas de un producto en una tienda. Lo que se hace es enviar al servidor un registro de SQL para que pueda insertarlo en la tabla correspondiente y en caso de consulta, se envía una consulta SQL para que el servidor muestre los registros que cumplen los parámetros de la consulta.

Salidas: En este caso, una vez que el servidor de Navision recibe el registro SQL del ordenador, debe procesar la información para poder hacer las pertinentes inserciones en

las tablas de SQL. A continuación el servidor emite un mensaje al cliente, que puede ser para informar que la función se ha hecho correctamente o que se ha producido un error a la hora de hacerlo.

Intercambio de ficheros: En este caso, si en alguna función existe la opción de importar los datos a Excel, el servidor le devuelve una hoja de Excel rellena con los datos.

4.2 Descripción a alto nivel de cada módulo

En este apartado se explica, de forma detallada, la funcionalidad de cada módulo. Además, se explican las principales funciones y algoritmos que se utilizan en cada módulo.

4.2.1 Almacén

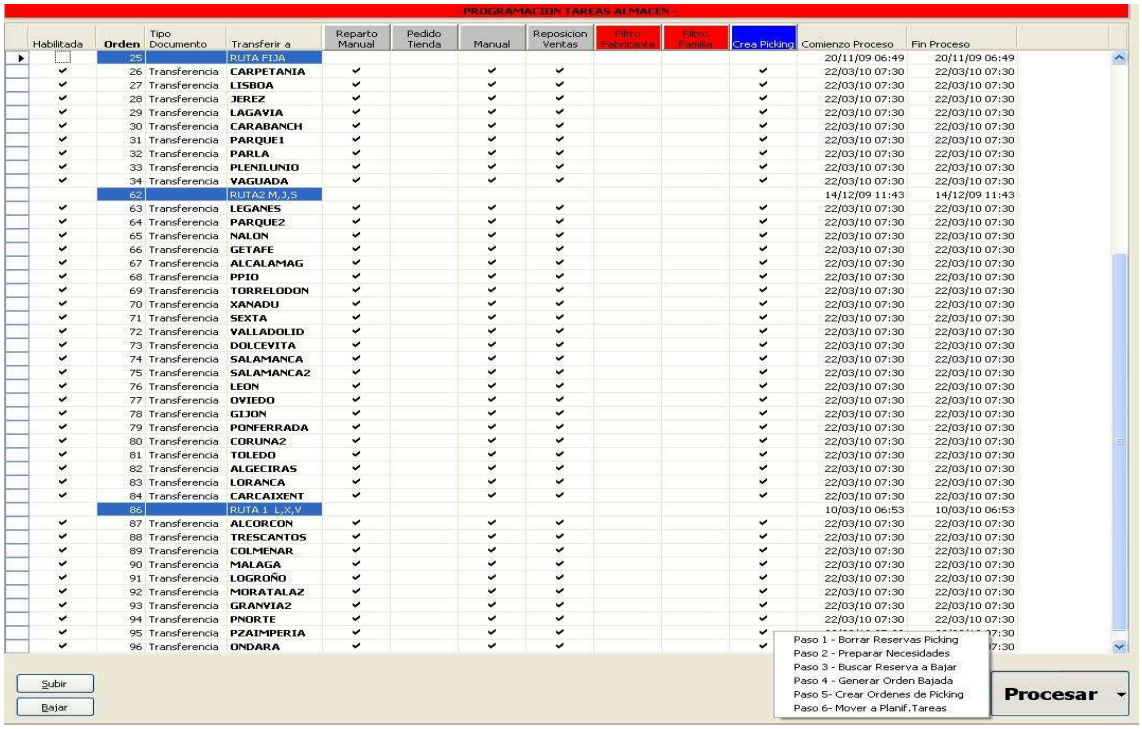


Ilustración 27 Formulario Generación Picking

Como se muestra en la ilustración 27, es donde se generan las órdenes de picking de cada una de las tiendas y se generan órdenes de bajada de palet para reaprovisionar la estantería.

Borrar Reservar Picking

En esta función lo que se hace es borrar la tabla de repartos de envíos, que es donde se van insertando las órdenes de picking que se van encontrando para la preparación de los pedidos. Entonces, una vez generadas las órdenes esos registros quedan inservibles. Por lo tanto, cuando se quieren volver a generar esas órdenes, se debe borrar esa tabla

Por otro lado, el sistema se recorre todas las líneas de envío y comprueba si alguna tiene alguna orden pendiente de picking. En caso de encontrarla, se debe borrar esa orden de picking debido a que el jefe de almacén cambia la prioridad de las tiendas al siguiente día y se debe quitar cualquier reserva pendiente del día anterior.

A continuación, se muestra el pseudocódigo de cada una de las funciones.

BorrarPicking

Pseudocódigo

```
TablaRepartos se inicializa
TablaRepartos se borran todos los registros

TablaEnvios se inicializa
Si TablaEnvios encuentra registros de la tabla entonces
    Repetir
        BorrarOrdenPicking(RegistroEnvios);
    Hasta que siguiente registro de la tabla  sea vacío
```

Código Original

```
rcdRepartoEnvio.RESET;
rcdRepartoEnvio.DELETEALL;
rcdLin.RESET;
IF rcdLin.FINDSET THEN
REPEAT
  BorrarPickingLineaEnvio(rcdLin."No.",rcdLin."Line No.");
UNTIL rcdLin.NEXT = 0;
```

BorrarOrdenPicking

Pseudocódigo

```
TablaLineasEnvios se inicializa
TablaLineasEnvios se filtra por número de envío del registro que se pasa por parámetro
TablaLineasEnvios se filtra por número de línea envío del registro que se pasa por parámetro
Si encuentras Registro TablaLineasEnvios entonces
        TablaLineasPicking se inicializa
        TablaLineasPicking filtra por tipo documento envio
        TablaLineasPicking filtra por numero de envió
        TablaLineasPicking filtra por numero línea de envío
```

```
Si encuentro registro TablasLineasPicking
        Borro Registro
```

Código Original

```
LinEnvio.RESET;
LinEnvio.SETRANGE(LinEnvio."No.",sEnvio);
LinEnvio.SETRANGE("Line No.",sLinEnvio);

IF LinEnvio.FINDFIRST THEN
  BEGIN
    LinPicking.RESET;
    LinPicking.SETRANGE("Whse. Document Type",LinPicking."Whse. Document
Type"::Shipment);
    LinPicking.SETRANGE("Whse. Document No.",LinEnvio."No.");
    LinPicking.SETRANGE("Whse. Document Line No.",sLinEnvio);
    IF LinPicking.FINDSET THEN
       LinPicking.DELETE;
```

Preparar Necesidades

En esta función lo que hace el sistema es recorrer, por cada tienda señalada, todas las transferencias que cumplan los requisitos marcados en el formulario para la tienda. Entonces si cumple el requisito, se recorre cada una de las líneas de la transferencia y comprueba si está metida en algún envío. En caso de no estar en un envío, se crea un nuevo envío para ir insertando las líneas de las transferencias que aún no estaban en un envío.

Este proceso se realiza para cada una de las tiendas y clientes que están señalados en el formulario.

A continuación, se muestra el pseudocódigo de las funciones.

ProcesarPedidos

PseudoCódigo

```
TablaTiendas se inicializa
TablaTiendas filtra por habilitadas

Si encuentra registros TablaTiendas entonces
        Repetir
                TablaSolicitudes se inicializa
                TablaSolicitudes filtra por tipo salida
                TablaSolicitudes filtra por almacén CENTRAL
                TablaSolicitudes filtra por Completamente Preparado FALSE
                TablaSolicitudes filtra por Estado del Documento Lanzado
                Si Tipo Documento Transferencia entonces
                        TablaSolicitudes filtra Documento Origen Transferencia
                        TablaSolicitudes filtra Tipo Destino Almacen
```

```
                TablaSolicitudes filtra Tipo Origen Línea Transferencia
                TablaSolicitudes filtra Destino Transferencia Nombre Tienda
        Si Tipo Documento Pedido Venta entonces
                TablaSolicitudes filtra Documento Origen Pedido Venta
                TablaSolicitudes filtra Tipo Destino Cliente
                TablaSolicitudes filtra Tipo Origen Línea Venta
                TablaSolicitudes filtra  Número Destino Número Pedido
        SI Tipo Documento Cliente entonces
                TablaSolicitudes filtra Documento Origen Pedido Venta
                TablaSolicitudes filtra Tipo Destino Cliente
                TablaSolicitudes filtra Tipo Origen Línea Venta
                TablaSolicitudes filtra  Número Destino Número Pedido
    Si encuentra registro TablaSolicitudes entonces
     Repetir
        Si Tipo Documento Transferencia entonces
            TablaTransfer se inicializa
            TablaTransfer filtra Numero Numero Origen TablaSolicitudes
            Si encuentra registro TablaTransfer entonces
                Si Tipo Transferencia esta dentro de los requisitos entonces
                    TablaLinTransfer se inicializa
                    TablaLinTransfer filtra Número por Numero TablaTransfer
                    TablaLinTransfer filtra Derivación 0
                    TablaLinTransfer filtra Cantidad mayor que 0
                    TablaLinTransfer filtra Cantidad Pendiente mayor que 0
                    Si encuentra registro entonces
                    Repetir
                        Comprueba si la línea esta en un envió
                       Si no esta en un envío entonces
                            Si no existe Cabecera Creada entonces
                                Crear Cabecera
                            Insertar Línea de Transferencia en el envío
                    Hasta registro TablaLinTransfer sea vacío
        Si Tipo Documento Pedido Venta entonces
            TablaVentas se inicializa
            TablaVentas filtra por tipo Pedido
            TablaVentas filtra Numero Numero Origen TablaSolicitudes
            Si encuentra registro TablaVentas entonces
                    TablaLinVenta se inicializa
                    TablaLinVenta filtra Número por Numero TablaVentas
                    TablaLinVenta filtra Tipo Producto
                    TablaLinVenta filtra Cantidad mayor que 0
                    TablaLinVenta filtra Cantidad Pendiente mayor que 0
                    Si encuentra registro entonces
                    Repetir
                        Comprueba si la línea esta en un envío
                       Si no esta en un envío entonces
                            Si no existe Cabecera Creada entonces
                                Crear Cabecera
                            Insertar Línea de Venta en el envío
                   Hasta registro TablaLinVenta sea vacío
          Hasta que registro TablaSolicitudes sea vacío
Hasta que registro TablaTiendas sea vacío
```

Código Original

```
Registro.SETRANGE(Habilitada,TRUE);

IF Registro.FINDSET(TRUE) THEN
REPEAT
   Registro."Comienzo Proceso":=CURRENTDATETIME;
   Registro.MODIFY;
   bCreaEnvio:=TRUE;
    WhseRqst.RESET;
   WhseRqst.SETRANGE(Type,WhseRqst.Type::Outbound);
   WhseRqst.SETRANGE("Location Code",'CENTRAL');
   WhseRqst.SETRANGE("Completely Handled",FALSE);
   WhseRqst.SETRANGE("DocumentStatus",WhseRqst."Document Status"::Released);
    // Para transferencias
   IF Registro."Tipo Documento"=Registro."Tipo Documento"::Transferencia THEN
   BEGIN
      WhseRqst.SETRANGE("SourceDocument",WhseRqst."Source
Document"::"Outbound Transfer");
      WhseRqst.SETRANGE("DestinationType",WhseRqst."Destination Type"::Location );
      WhseRqst.SETRANGE("Source Type",DATABASE::"Transfer Line");
      WhseRqst.SETRANGE("Destino Transferencia",Registro."Transferir a");
   END;

   // Para Pedidos de venta
   IF Registro."Tipo Documento"=Registro."Tipo Documento"::"Pedido Venta" THEN
   BEGIN
    WhseRqst.SETRANGE("Source Document",WhseRqst."Source Document"::"Sales
Order");
     WhseRqst.SETRANGE("DestinationType",WhseRqst."Destination Type"::Customer );
     WhseRqst.SETRANGE("Source Type",DATABASE::"Sales Line");
     WhseRqst.SETRANGE("Source No.",Registro."Transferir a");
   END;

   // Para Clientes enteros
   IF Registro."Tipo Documento"=Registro."Tipo Documento"::Cliente THEN
   BEGIN
    WhseRqst.SETRANGE("Source Document",WhseRqst."Source Document"::"Sales
Order");
     WhseRqst.SETRANGE("DestinationType",WhseRqst."Destination Type"::Customer );
     WhseRqst.SETRANGE("Source Type",DATABASE::"Sales Line");
     WhseRqst.SETRANGE("Destination No.",Registro."Transferir a");
   END;
   // Recorro todas las lineas de los documentos de origen que me voy a traer
   IF WhseRqst.FINDSET THEN
   REPEAT

//------------------------- TRANSFERENCIAS
    IF WhseRqst."Source Document"= WhseRqst."Source Document"::"Outbound Transfer"
THEN
     BEGIN
      CabTransferencia.RESET;
      CabTransferencia.SETRANGE("No.",WhseRqst."Source No.");
```

```
      IF CabTransferencia.FINDFIRST THEN
      BEGIN
        // Validamos que sea del tipo correcto
        bProcesar:=FALSE;
         // Pedidos manuales desde Navision
         IF(CabTransferencia."TipoTransferencia"=CabTransferencia."Tipo
Transferencia"::"Manual ") AND (Registro.Manual) THEN
           bProcesar:=TRUE;

        // Pedidos Manuales desde tienda
        IF(CabTransferencia."TipoTransferencia"=CabTransferencia."Tipo
Transferencia"::"Pedido Tienda") AND (Registro."Pedido Tienda") THEN
          bProcesar:=TRUE;

        // Pedidos desde Repartos Manuales
        IF(CabTransferencia."TipoTransferencia"=CabTransferencia."Tipo
Transferencia"::"Reparto Manual") AND (Registro."Reparto Manual") THEN
          bProcesar:=TRUE;

        // Pedidos Manuales Repartos Cross-Docking
        IF(CabTransferencia."TipoTransferencia"=CabTransferencia."Tipo
Transferencia"::"Reparto Cross-Docking") AND (Registro."Reparto Cross-Docking") THEN
          bProcesar:=TRUE;



         // Pedidos de reposición Ventas
        IF(CabTransferencia."TipoTransferencia"=CabTransferencia."Tipo
Transferencia"::"Proceso Ventas") AND (Registro."Reposicion Ventas") THEN
          bProcesar:=TRUE;

        IF bProcesar THEN
        BEGIN
          LinTransferencia.RESET;
          LinTransferencia.SETRANGE("Document No.",CabTransferencia."No.");
          LinTransferencia.SETRANGE("Derived From Line No.",0);
          LinTransferencia.SETFILTER(Quantity,'>0');
          LinTransferencia.SETFILTER("Outstanding Quantity",'>0');

       IF LinTransferencia.FINDSET THEN
       BEGIN
         REPEAT
           IF WhseActivityCreate.CheckIfFromTransLine2ShptLine(LinTransferencia) THEN
           BEGIN
             IF bCreaEnvio THEN
             BEGIN
              CreaEnvio(CabEnvio);
              bCreaEnvio:=FALSE;
             END;

            WhseActivityCreate.FromTransLine2ShptLine(CabEnvio,LinTransferencia);
           END;
         UNTIL LinTransferencia.NEXT = 0;
        END;
```

```
      END;
    END;


//-------------------------- PEDIDOS DE VENTA
    IF WhseRqst."Source Document"= WhseRqst."Source Document"::"Sales Order" THEN
    BEGIN
      CabVenta.RESET;
      CabVenta.SETRANGE("Document Type",CabVenta."Document Type"::Order);
      CabVenta.SETRANGE("No.",WhseRqst."Source No.");
      IF CabVenta.FINDFIRST THEN
      BEGIN
        LinVenta.RESET;
        LinVenta.SETRANGE("Document Type",LinVenta."Document Type"::Order);
        LinVenta.SETRANGE("Document No.",CabVenta."No.");
        LinVenta.SETRANGE(LinVenta.Type,LinVenta.Type::Item);
        LinVenta.SETFILTER(Quantity,'>0');
        LinVenta.SETFILTER("Outstanding Quantity",'>0');
        IF LinVenta.FINDSET THEN
        REPEAT
          IF WhseActivityCreate.CheckIfFromSalesLine2ShptLine(LinVenta) THEN
          BEGIN
            IF bCreaEnvio THEN
            BEGIN
              NumTotEnvio+=1;
              CreaEnvio(CabEnvio);
              bCreaEnvio:=FALSE;
            END;

            WhseActivityCreate.FromSalesLine2ShptLine(CabEnvio,LinVenta);

        UNTIL LinVenta.NEXT = 0;
      END;
    END;
    UNTIL (WhseRqst.NEXT = 0);
UNTIL Registro.NEXT = 0;
```

Crear Órdenes de Picking

En esta función lo que hace el sistema es recorrer las líneas de envío que aún no han sido ni preparadas ni buscadas en estantería para preparar. Dependiendo de que tipo sea el registro se recorre las líneas del envío que cumplan dicho filtro. En este caso, puede ser que le toque recorrer una tienda o un cliente.

En caso de ser tienda el sistema realiza lo siguiente:

En primer lugar, busca si para esa tienda existe ya un picking creado. En caso de existir, para este nuevo picking se incrementa el número de documento externo del picking encontrado. Por otro lado, en caso de que no haya picking generado, se asigna el número 01 al campo documento externo del picking que se va a generar.

A continuación, el sistema se va recorriendo cada una de las líneas del envío y realiza para cada una de ellas lo siguiente:

En primer lugar, comprueba que la línea de envío está dentro de los filtros que se han configurado en la pantalla inicial. En caso de cumplir los requisitos, el sistema busca en todas las ubicaciones de picking donde esté ubicado el producto.

En cada una de ellas comprueba si hay cantidad disponible o ya está toda reservada. En caso de haber cantidad disponible, se comprueba si se asigna toda la cantidad disponible, ya que la cantidad necesaria es mayor o se le asigna la cantidad necesaria. Una vez calculado, se genera una línea de reparto donde se indica de qué ubicación de picking se va a preparar el pedido y la cantidad que se cogerá de dicha ubicación, para que luego se puedan generar las pertinentes órdenes de picking. Por último, se comprueba si queda pendiente aún cantidad por asignar a la línea de envío. En caso de quedar cantidad pendiente, se debe mirar en las demás ubicaciones de picking si es posible asignar cantidad a esa línea, proceso explicado anteriormente.

Una vez acabado el anterior proceso, en cada una de las líneas de reparto generadas se asigna el número de documento externo calculado anteriormente. Entonces, una vez hecho ese proceso lo único que queda es generar la hoja de picking.

En caso de ser cliente el sistema realiza lo siguiente:

En primer lugar, busca si para ese cliente existe ya un picking creado. En caso de existir, para este nuevo picking se incrementa el número de documento externo del picking encontrado. Por otro lado, en caso de que no haya picking generado, se asigna el número 01 al campo documento externo del picking que se va a generar.

A continuación, el sistema se va recorriendo cada una de las líneas del envío y realiza para cada una de ellas lo siguiente:

En primer lugar, comprueba que la línea de envío está dentro de los filtros que se han configurado en la pantalla inicial. En caso de cumplir los requisitos, el sistema busca en todas las ubicaciones de picking donde esté ubicado el producto.

En cada una de ella comprueba si hay cantidad disponible o ya está toda reservada. En caso de haber cantidad disponible, se comprueba si se le asigna toda la cantidad disponible, ya que la cantidad necesaria es mayor o se le asigna la cantidad necesaria. Una vez calculado, se genera una línea de reparto donde se indica de qué ubicación de picking se va a preparar el pedido y la cantidad que se cogerá de dicha ubicación, para que luego se puedan generar las pertinentes órdenes de picking. Por

último, se comprueba si queda pendiente aún cantidad por asignar a la línea del envío. En caso de quedar cantidad pendiente, se debe mirar en las demás ubicaciones de picking si es posible asignar cantidad a esa línea, proceso explicado anteriormente.

Una vez acabado el anterior proceso, en cada una de las líneas de reparto generadas se asigna el número de documento externo calculado anteriormente. Entonces, una vez hecho ese proceso lo único que queda es generar la hoja de picking.

ProcesarPicking

PseudoCódigo

```
TablaTiendas se inicializa
TablaTiendas filtra por habilitadas
Si encuentra registro TablaTiendas entonces
    Repetir
        Caso Tipo Documento
              Transferencia: CrearPickingTienda (Registro)
             Cliente: CrearPickingCliente (Registro)
             Pedido Venta: CrearPickingCliente (Registro)
    Hasta siguiente registro tabla tiendas sea vacío
```

Código Original

```
Registro.SETCURRENTKEY(Orden);
Registro.SETRANGE(Habilitada,TRUE);

IF Registro.FINDSET(TRUE) THEN
REPEAT
  CASE Registro."Tipo Documento" OF
      Registro."Tipo Documento"::Transferencia:   CreaPickingTotalTiendas(Registro);
      Registro."Tipo Documento"::"Pedido Venta":  CreaPickingTotalPedido(Registro);
      Registro."Tipo Documento"::Cliente:         CreaPickingTotalPedido(Registro);
  END;
  COMMIT;
UNTIL Registro.NEXT=0;
```

CrearPickingTotalTiendas

PseudoCódigo

```
numDocAgrupado= Registro.NombreTienda+’0’
CabeceraPicking se inicializa
CabeceraPicking se filtra por numDocAgrupado*
Si encuentra registro CabeceraPicking entonces
    numDocAgrupado=incrementar(numDocAgrupado)
sino
    numDocAgrupado:=numDocAgrupado+'000001';

LineasEnvio se inicializa
LineasEnvio se filtra tipoDestino sea almacén
```

LineasEnvio se filtra numerodestino sea registro.NombreTienda
LineasEnvio se filtra sinreparto sea FALSE
LineasEnvio se filtra cantidadpendiente sea mayor que 0

Si encuentra registro LineasEnvio entonces
Repetir
Comprueba si la línea cumple los filtros configurados
Si cumple requisitos entonces
GenerarRepartoLineaEnvio(LineasEnvio.NumeroEnvio,
LineasEnvio.NumeroLineaEnvio,1,TRUE);
LineasReparto se inicializa
LineasReparto se filtra numeroenvio por LineasEnvio.NumeroEnvio
LineasReparto se filtra numerolineaenvio por LineasEnvio.NumeroLineaEnvio
Si encuentra registro LineasReparto entonces
Se modifica LineasReparto.NumeroDocumentoAgrupado por numDocAgrupado
Hasta que siguiente registro LineasEnvio sea vacío
CrearPickingAgrupado(numDocAgrupado);

Código Original

```
numDocAgrupado:=PARRegistro."Transferir a"+'0';
rcdPicking.RESET;
rcdPicking.SETRANGE(Type,rcdPicking.Type::Pick);
rcdPicking.SETFILTER("External Document No.",'%1',numDocAgrupado+'*');
IF rcdPicking.FINDLAST THEN
  numDocAgrupado:=INCSTR(rcdPicking."External Document No.")
ELSE
  numDocAgrupado:=numDocAgrupado+'000001';

rcdEnvio.RESET;
rcdEnvio.SETCURRENTKEY("No.","Destination Type","Destination No.","Item No.");
rcdEnvio.SETRANGE("Destination Type",rcdEnvio."Destination Type"::Location);
rcdEnvio.SETRANGE("Destination No.",PARRegistro."Transferir a");
rcdEnvio.SETRANGE("Sin Reparto",FALSE);
rcdEnvio.SETFILTER(rcdEnvio."Qty. Outstanding (Base)",'>0');
IF rcdEnvio.FINDSET THEN
REPEAT
    bProcesar:=TRUE;
// Comprobamos filtros de familia y fabricante
 IF PARRegistro."Filtro Familia" <> '' THEN
 BEGIN
   bProcesar:=FALSE;
   rcdItem.RESET;
   rcdItem.SETRANGE("No.",rcdEnvio."Item No.");
   IF rcdItem.FINDFIRST THEN
     IF rcdItem."Global Dimension 1 Code" = PARRegistro."Filtro Familia" THEN
       bProcesar:=TRUE;
 END;

 IF PARRegistro."Filtro Fabricante" <> '' THEN
 BEGIN
  bProcesar:=FALSE;
  rcdItem.RESET;
```

```
    rcdItem.SETRANGE("No.",rcdEnvio."Item No.");
    IF rcdItem.FINDFIRST THEN
      IF rcdItem."Manufacturer Code" = PARRegistro."Filtro Fabricante" THEN
        bProcesar:=TRUE;
  END;

  // Si ha filtrado por algún origen en concreto, verificamos que sea correcto
  IF (bProcesar) AND
     ((PARRegistro.Manual) OR (PARRegistro."Pedido Tienda") OR (PARRegistro."Reposicion
Ventas") OR
     (PARRegistro."Reparto Manual")) THEN
  BEGIN
    bProcesar:=FALSE;
    // Comprobamos que el origen sea válido
    CabTransferencia.RESET;
    CabTransferencia.SETRANGE("No.",rcdEnvio."Source No.");
    IF CabTransferencia.FINDFIRST THEN
    BEGIN
      // Pedidos manuales desde Navision
     IF(CabTransferencia."TipoTransferencia"=CabTransferencia."TipoTransferencia"::"Manual
") AND
        (PARRegistro.Manual) THEN
        bProcesar:=TRUE;
      // Pedidos Manuales desde tienda
      IF(CabTransferencia."TipoTransferencia"=CabTransferencia."TipoTransferencia"::"Pedido
Tienda") AND
        (PARRegistro."Pedido Tienda") THEN
        bProcesar:=TRUE;
      // Pedidos desde Repartos Manuales
      IF(CabTransferencia."TipoTransferencia"=CabTransferencia."TipoTransferencia"::"Reparto
Manual") AND
        (PARRegistro."Reparto Manual") THEN
        bProcesar:=TRUE;
      // Pedidos de reposición Ventas
      IF(CabTransferencia."TipoTransferencia"=CabTransferencia."TipoTransferencia"::"Proceso
Ventas") AND
        (PARRegistro."Reposicion Ventas") THEN
        bProcesar:=TRUE;
    END;
  END;


 IF bProcesar THEN
  BEGIN
    GenerarRepartoLineaEnvio(rcdEnvio."No.",rcdEnvio."Line No.",1,TRUE);
     rcdReparto.RESET;
    rcdReparto.SETRANGE("No Envio",rcdEnvio."No.");
    rcdReparto.SETRANGE("No Linea Envio",rcdEnvio."Line No.");
    IF rcdReparto.FINDFIRST THEN
       rcdReparto.MODIFYALL("No Documento Agrupado",numDocAgrupado);
      COMMIT;
   END;
UNTIL rcdEnvio.NEXT = 0;
CrearPickingAgrupado(numDocAgrupado,Ventana);
```

CrearPickingTotalPedido

PseudoCódigo

```
numDocAgrupado= Registro.NombreTienda+’0’
CabeceraPicking se inicializa
CabeceraPicking se filtra por numDocAgrupado*
Si encuentra registro CabeceraPicking entonces
    numDocAgrupado=incrementar(numDocAgrupado)
sino
    numDocAgrupado:=numDocAgrupado+'000001';

LineasEnvio se inicializa
Si TipoDocumento= Pedido Venta entonces
       LineasEnvio se filtra DocumentoOrigen sea Pedido Venta
       LineasEnvio se filtra numeroorigen sea registro.NombreTienda
sino
       LineasEnvio se filtra tipodestino sea Cliente
       LineasEnvio se filtra numerodestino sea registro.NombreTienda

LineasEnvio se filtra sinreparto sea FALSE
LineasEnvio se filtra cantidadpendiente sea mayor que 0
Si encuentra registro LineasEnvio entonces
 Repetir
   Comprueba si la línea cumple los filtros configurados
   Si cumple requisitos entonces
        GenerarRepartoLineaEnvio(LineasEnvio.NumeroEnvio,
LineasEnvio.NumeroLineaEnvio,1,TRUE);
   LineasReparto se inicializa
   LineasReparto se filtra numeroenvio por LineasEnvio.NumeroEnvio
   LineasReparto se filtra numerolineaenvio por LineasEnvio.NumeroLineaEnvio
   Si encuentra registro LineasReparto entonces
      Se modifica LineasReparto.NumeroDocumentoAgrupado por numDocAgrupado
Hasta que siguiente registro LineasEnvio sea vacío
CrearPickingAgrupado(numDocAgrupado);
```

Código Original

```
numDocAgrupado:=PARRegistro."Transferir a";

rcdPicking.RESET;
rcdPicking.SETRANGE(Type,rcdPicking.Type::Pick);
rcdPicking.SETFILTER("External Document No.",'%1',numDocAgrupado+'*');
IF rcdPicking.FINDLAST THEN
  numDocAgrupado:=INCSTR(rcdPicking."External Document No.")
ELSE
  numDocAgrupado:=numDocAgrupado+'0000001';

rcdEnvio.RESET;
IF PARRegistro."Tipo Documento" = PARRegistro."Tipo Documento"::"Pedido Venta" THEN
BEGIN
  rcdEnvio.SETCURRENTKEY("No.","Source Document","Source No.");
  rcdEnvio.SETRANGE("SourceDocument",rcdEnvio."SourceDocument"::"Sales Order");
```

```
  rcdEnvio.SETRANGE("Source No.",PARRegistro."Transferir a");
END
ELSE
BEGIN
  rcdEnvio.SETCURRENTKEY("No.","Destination Type","Destination No.","Item No.");
  rcdEnvio.SETRANGE("Destination Type",rcdEnvio."Destination Type"::Customer);
  rcdEnvio.SETRANGE("Destination No.",PARRegistro."Transferir a");
END;

rcdEnvio.SETRANGE("Sin Reparto",FALSE);
rcdEnvio.SETFILTER(rcdEnvio."Qty. Outstanding (Base)",'>0');

IF rcdEnvio.FINDSET THEN
REPEAT
 bProcesar:=TRUE;

 IF PARRegistro."Filtro Familia" <> '' THEN
 BEGIN
   bProcesar:=FALSE;
   rcdItem.RESET;
   rcdItem.SETRANGE("No.",rcdEnvio."Item No.");
   IF rcdItem.FINDFIRST THEN
     IF rcdItem."Global Dimension 1 Code" = PARRegistro."Filtro Familia" THEN
       bProcesar:=TRUE;
 END;

 IF PARRegistro."Filtro Fabricante" <> '' THEN
 BEGIN
  bProcesar:=FALSE;
  rcdItem.RESET;
  rcdItem.SETRANGE("No.",rcdEnvio."Item No.");
  IF rcdItem.FINDFIRST THEN
    IF rcdItem."Manufacturer Code" = PARRegistro."Filtro Fabricante" THEN
      bProcesar:=TRUE;
 END;

 IF bProcesar THEN
 BEGIN
  FuncionesEnvio.GenerarRepartoLineaEnvio(rcdEnvio."No.",rcdEnvio."Line No.",1,TRUE);

  rcdReparto.RESET;
  rcdReparto.SETRANGE("No Envio",rcdEnvio."No.");
  rcdReparto.SETRANGE("No Linea Envio",rcdEnvio."Line No.");
  IF rcdReparto.FINDFIRST THEN
    rcdReparto.MODIFYALL("No Documento Agrupado",numDocAgrupado);
  COMMIT;
 END;
UNTIL rcdEnvio.NEXT = 0;
 FuncionesEnvio.CrearPickingAgrupado(numDocAgrupado,Ventana);
```

GenerarRepartoLineaEnvio

PseudoCódigo

TablaReparto se inicializa
TablaReparto se filtra numeroenvio por sEnvio
TablaReparto se filtra numerolineaenvio por NLinenvio
Se borra los registros que estén dentro del filtro

CabeceraEnvio se inicializa
CabeceraEnvio se filtra numero por sEnvio
Si no encuentra registro entonces
ERROR (‘No existe envío’);

LineasEnvio se inicializa
LineasEnvio se filtra número por CabeceraEnvio.Numero
LineasEnvio se filtra numerolinea por NLinEnvio
LineasEnvio se filtra sinreparto sea FALSE

Si encuentra registros LineasEnvio entonces
Repetir
LineasEnvio se calcula Cantidad Localizada (Base), Cantidad Picking (Base)
dcantidad= Cantidad Pendiente- Cantidad Localizada- Cantidad Preparada- Cantidad Picking
TablaProducto se inicializa
TablaProducto se filtra número por LineasEnvio.NumeroProducto
Si no se encuentra registro TablaProducto entonces
ERROR (‘No existe producto’);
Si dcantidad>0 y picking entonces
BuscarEnZonaPicking(EnvioLineas."No.",EnvioLineas."LineNo.", EnvioLineas."Item No.", dCantidad, 2);
LineasEnvio se inicializa
LineasEnvio se filtra número por CabeceraEnvio.Numero
LineasEnvio se filtra numerolinea por NLinEnvio
Si encuentra registros LineasEnvio entonces
LineasEnvio se calcula Cantidad Localizada (Base), Cantidad Picking (Base)
dcantidad= Cantidad Pendiente- Cantidad Localizada- Cantidad Preparada- Cantidad Picking
Hasta que registro LineasEnvio sea vacío

Código Original

```
Reparto.RESET;
Reparto.SETRANGE(Reparto."No Envio", sEnvio);
Reparto.SETRANGE(Reparto."No Linea Envio", NLinenvio);
Reparto.DELETEALL;

Envio.RESET;
Envio.SETRANGE("No.",sEnvio);
IF NOT Envio.FINDFIRST THEN
  ERROR ('No Encuentro el Envio %1 ',sEnvio);

EnvioLineas.RESET;
EnvioLineas.SETRANGE(EnvioLineas."No.", Envio."No.");
EnvioLineas.SETRANGE(EnvioLineas."Line No.", NLinenvio);
EnvioLineas.SETRANGE(EnvioLineas."Sin Reparto", FALSE);

IF EnvioLineas.FINDSET THEN
REPEAT
```

```
   EnvioLineas.CALCFIELDS("Cantidad Localizada (Base)","Pick Qty. (Base)");
   dCantidad := EnvioLineas."Qty. Outstanding (Base)" - EnvioLineas."Cantidad Localizada
(Base)"- EnvioLineas."Qty. Picked (Base)" - EnvioLineas."Pick Qty. (Base)";
   Producto.RESET;
   Producto.SETRANGE("No.",EnvioLineas."Item No.");
   IF NOT Producto.FINDFIRST THEN
     ERROR('No encuentro el producto %1',EnvioLineas."Item No.");

   IF (dCantidad > 0) AND (bSoloPicking) THEN
   BEGIN
     BuscarEnZonaPicking(EnvioLineas."No.",EnvioLineas."LineNo.", EnvioLineas."Item No.",
dCantidad, 2);

     EnvioLineasAux.RESET;
     EnvioLineasAux.SETRANGE("No.",EnvioLineas."No.");
     EnvioLineasAux.SETRANGE("Line No.",EnvioLineas."Line No.");
     IF EnvioLineasAux.FINDFIRST THEN
       EnvioLineasAux.CALCFIELDS("Cantidad Localizada (Base)","Pick Qty. (Base)");

     dCantidad:=EnvioLineasAux."Qty.Outstanding(Base)" - EnvioLineasAux."Cantidad
Localizada (Base)"- EnvioLineasAux."Qty. Picked (Base)" - EnvioLineasAux."Pick Qty.
(Base)";
   END;
UNTIL EnvioLineas.NEXT = 0;
```

BuscarEnZonaPicking

PseudoCódigo

```
Dcantidadpendiente= dcantidad;
Dcantidadpicking=CalcularCantidadPicking(sproducto);
Si (DcantidadPicking>Dcantidadpendiente) O (iteración=2 Y Dcantidadpicking>0)
    CantidadDisponible=0;
    CantidadIntroducir=0;
    TablaContenidos se inicializa
    TablaContenidos se filtra Codigo Ubicación<>'ISLA'Y 'RTOMASIVO'
    TablaContenidos se filtra númeroproducto sea sproducto
    TablaContenidos se filtra Codigo Zona sea PICKING
    Si se encuentra registros TablaContenidos entonces
    REPETIR
         TablaContenidos se calcula Cantidad, Cantidad Picking y Cantidad Reparto
         CantidadDisponible=Cantidad-Cantidad Picking-Cantidad Reparto
         IF CantidadDisponible>0 entonces
             Si CantidadDisponible>Dcantidadpendiente entonces
                 CantidadIntroducir=Dcantidadpendiente
             Sino
                 CantidadIntroducir=CantidadDisponible
        CrearLineaRepartoEnvio(sEnvio,iLineaEnvio,sProducto,CantidadIntroducir,
'CENTRAL',TablaContenidos."Zone Code", TablaContenidos."Bin Code", '', 0);
         Dcantidadpendiente= Dcantidadpendiente- CantidadIntroducir;
         CantidadDisponible=0;
         CantidadIntroducir=0;
   HASTA registro TablaContenidos sea vacío O Dcantidadpendiente=0;
```

Código Original

```
dCantPendiente := iCantidad;
dCantPicking := CalcExistenciasDispPicking(sProducto);
IF (dCantPicking >= dCantPendiente) OR ((iIteracion = 2) AND (dCantPicking > 0)) THEN
BEGIN
   dCantDisponible := 0;
   dCantIntroducir := 0;
    BinContent.RESET;
   BinContent.SETRANGE(BinContent."Location Code", ‘CENTRAL’);
   BinContent.SETFILTER(BinContent."Bin Code", '<>ISLA&<>RTOMASIVO');
   BinContent.SETRANGE(BinContent."Item No.", sProducto);
   BinContent.SETRANGE(BinContent."Zone Code", ‘PICKING’);
   IF BinContent.FINDSET THEN
   REPEAT
      BinContent.CALCFIELDS(Quantity, "Pick Qty.", "Cantidad Reparto Envio");
      dCantDisponible := BinContent.Quantity - BinContent."Pick Qty." - BinContent."Cantidad
Reparto Envio";
      IF (dCantDisponible > 0) THEN
      BEGIN
         IF (dCantDisponible > dCantPendiente) THEN
            dCantIntroducir := dCantPendiente
         ELSE
            dCantIntroducir := dCantDisponible;
         CrearLineaRepartoEnvio(sEnvio, iLineaEnvio, sProducto, dCantIntroducir,
BinContent."Location Code",BinContent."Zone Code", BinContent."Bin Code", '', 0);
         dCantPendiente := dCantPendiente - dCantIntroducir;
         dCantDisponible := 0;
         dCantIntroducir := 0;
      END;

   UNTIL (BinContent.NEXT = 0) OR (dCantPendiente = 0);
```

CalcExistenciasDispPicking

PseudoCódigo

```
TablaProducto se inicializa
TablaProducto se filtra número por sproducto
Si no encuentra registros TablaProducto entonces
    ERROR (‘No existe producto’);
TablaProductos se calcula Existencias Picking
TablaActividad se inicializa
TablaActividad se filtra tipo actividad sea Picking
TablaActividad se filtra tipo accion sea traer
TablaActividad se filtra codigo zona sea PICKING
TablaActividad se filtra destino línea sea Ubicación
TablaActividad se filtra numero producto sea sProducto
TablaActividad se filtra Trilateral sea falso
TablaActividad se filtra Reaprovisionamiento sea falso
Dcantidad=0;
Si encuentra registros TablaActividad entonces
REPETIR
     Dcantidad=Dcantidad+ TablaActividad.CantidadPendiente
HASTA que siguiente registro TablaActividad sea nulo
Se devuelve TablaProducto.Existencias Picking-Dcantidad;
```

Código Original

```
ProductoAux.RESET;
ProductoAux.SETRANGE("No.",sProducto);
IF NOT ProductoAux.FINDFIRST THEN
  ERROR ('El producto %1 No exite',sProducto);
ProductoAux.CALCFIELDS("Existencias Picking");

WhoseActivLine.RESET;
WhoseActivLine.SETCURRENTKEY("Action  Type",Trilateral,Reaprovisionamiento,"Destino
Linea",Prioridad,Sector,"Bin Code");
WhoseActivLine.SETRANGE(WhoseActivLine."ActivityType",      WhoseActivLine."Activity
Type"::Pick);
WhoseActivLine.SETRANGE(WhoseActivLine."ActionType",         WhoseActivLine."Action
Type"::Take);
WhoseActivLine.SETRANGE(WhoseActivLine."Zone Code", 'PICKING');
WhoseActivLine.SETRANGE(WhoseActivLine."DestinoLinea",      WhoseActivLine."Destino
Linea"::Ubicación);
WhoseActivLine.SETRANGE(WhoseActivLine."Item No.", sProducto);
WhoseActivLine.SETRANGE(WhoseActivLine.Trilateral, FALSE);
WhoseActivLine.SETRANGE(WhoseActivLine.Reaprovisionamiento, FALSE);
dCantidad := 0;
IF (WhoseActivLine.FINDSET) THEN
REPEAT
  dCantidad += WhoseActivLine."Qty. Outstanding (Base)";
UNTIL WhoseActivLine.NEXT = 0;

EXIT(ProductoAux."Existencias Picking" - dCantidad);
```

CrearLineaRepartoEnvio

PseudoCódigo

```
TablaProducto se inicializa
TablaProducto se filtra número por sProducto
Si no se encuentra registros TablaProducto entonces
    ERROR (‘No existe producto’);
TablaRepartos se inicializa
TablaRepartos se filtra número envio por sEnvío
TablaRepartos se filtra número línea envío por iLineaEnvío
Si encuentra ultimo registro TablaRepartos entonces
    NumLineaReparto= TablaReparto. NumLineaReparto+10000;
Sino
    NumLineaReparto=10000;
TablaRepartos se inicializa
TablaRepartos.NumEnvío=sEnvío;
TablaRepartos.No Linea Envío=iLineaEnvío

TablaRepartos.Cod Producto= sProducto;
TablaRepartos.Cantidad = iCantidad;
TablaRepartos.Cantidad (base) = iCantidad;
TablaRepartos.Cod Ud Medida= TablaProducto.Base Unit of Measure;
TablaRepartos.Cant Por Ud Medida= 1;
```

```
TablaRepartos.Estado = TablaRepartos.Estado::" ";
TablaRepartos.Almacen Origen= sAlmacen;
TablaRepartos.Zona Origen= sZona;
TablaRepartos.Ubicacion Origen= sUbicacion;
Si  sBulto <>'' entonces
   TablaRepartos.Proceso Envio= TablaRepartos.Proceso Envio::Bulto;
   TablaRepartos.Nº Bulto= sBulto;
   TablaRepartos.No Linea Bulto= iLineaBulto;
Sino
   TablaRepartos.Proceso Envio= TablaRepartos.Proceso Envio::Picking;
Se inserta el registro;
```

Código Original

```
Producto.RESET;
Producto.SETRANGE("No.",sProducto);
IF NOT Producto.FINDFIRST THEN
   ERROR ('NO existe el producto %1 ',sProducto) ;
Reparto.RESET;
Reparto.SETRANGE("No Envio",sEnvio);
Reparto.SETRANGE("No Linea Envio",iLineaEnvio);
IF Reparto.FINDLAST THEN
BEGIN
    NumLinReparto := Reparto."No Linea Reparto" + 10000;
END ELSE BEGIN
    NumLinReparto := 10000;
END;
Reparto.INIT;
Reparto."No Envio" := sEnvio;
Reparto."No Linea Envio" := iLineaEnvio;
Reparto."No Linea Reparto" := NumLinReparto;
Reparto."Cod Producto" := sProducto;
Reparto.Cantidad := iCantidad;
Reparto."Cantidad (base)" := iCantidad;
Reparto."Cod Ud Medida" := Producto."Base Unit of Measure";
Reparto."Cant Por Ud Medida" := 1;
Reparto.Estado := Reparto.Estado::" ";
Reparto."Almacen Origen" := sAlmacen;
Reparto."Zona Origen" := sZona;
Reparto."Ubicacion Origen" := sUbicacion;
IF (sBulto <> '') THEN
BEGIN
   Reparto."Proceso Envio" := Reparto."Proceso Envio"::Bulto;
   Reparto."Nº Bulto" := sBulto;
   Reparto."No Linea Bulto" := iLineaBulto;
END ELSE BEGIN
   Reparto."Proceso Envio" := Reparto."Proceso Envio"::Picking;
END;
Reparto.INSERT;
```

CrearPickingAgrupado

PseudoCódigo

numLinea := 10000;
TablaCabActividad se inicializa
TablaCabActividad se filtra Tipo sea picking
TablaCabActividad se filtra Nº Documento Externo sea NumDocAgrupado;
Si no se encuentra registros TablaCabActividad entonces
  TablaConfigActividad se inicializa
  TablaConfigActividad se situa en el primer registro
  TablaCabActividad se inicializa;
  TablaCabActividad.Tipo = TablaCabActividad.Tipo::Pick;
  TablaCabActividad.Codigo Almacén = 'CENTRAL';
  TablaCabActividad.MetodoOrdenación= TablaCabActividad.MetodoOrdenación::Action Type;
  TablaCabActividad.Documento Externo = NumDocAgrupado;
  TablaCabActividad. Numero Serie = TablaConfigActividad.Numero Serie
  Se inserta el registro

TablaActividad se inicializa;
TablaActividad se filtra tipo actividad Picking
TablaActividad se filtra número por TablaCabActividad. Numero
Si se encuentra el ultimo registro TablaActividad
  numLinea := TablaActividad.NumeroLinea + 10000;

TablaReparto se inicializa;
TablaReparto se filtra Documento Agrupado por NumDocAgrupado;
TablaReparto se filtra Estado por " ";

Si se encuentra registros TablaReparto entonces
Repetir
  LineasEnvio se inicializa;
  LineasEnvio se filtra número por TablaReparto.No Envio;
  LineasEnvio se filtra número línea por TablaReparto.No Linea Envio;
  LineasEnvio se posiciona en el primer registro

  TablaActividad se inicializa;
  TablaActividad.Tipo Actividad = TablaCabActividad.Tipo;
  TablaActividad.Numero= TablaCabActividad.Numero;
  TablaActividad = numLinea;
  TablaActividad.Tipo Accion= TablaActividad.Tipo Accion ::Take;
  TablaActividad.Tipo Origen= LineasEnvio.Tipo Origen;
  TablaActividad.Subtipo Origen= LineasEnvio. Subtipo Origen;
  TablaActividad.NumeroOrigen= LineasEnvio.NumeroOrigen;
  TablaActividad.NumeroLineaOrigen= LineasEnvio. NumeroLineaOrigen;
  TablaActividad. NumeroSubLineaOrigen = TablaReparto.NumeroLineaReparto;
  TablaActividad.DocumentoOrigen= LineasEnvio.DocumentoOrigen;
  TablaActividad.CodigoAlmacen= LineasEnvio.CodigoAlmacen;
  TablaActividad.NumeroEstante= LineasEnvio.NumeroEstante;
  TablaActividad.NumeroProducto= LineasEnvio. NumeroProducto;
  TablaActividad.CodigoVariante= LineasEnvio. CodigoVariante;
  TablaActividad.UnidadMedida= LineasEnvio. UnidadMedida;
  TablaActividad.CantidadUnidadMedida= LineasEnvio. CantidadUnidadMedida;
  TablaActividad.Descripcion = LineasEnvio. Descripcion;
  TablaActividad. Descripcion2 = LineasEnvio. Descripcion2;
  TablaActividad.FechaVencimiento= LineasEnvio.FechaVencimiento;
  TablaActividad.FechaInicio= LineasEnvio.FechaEnvio;

```
    TablaActividad.TipoDestino= LineasEnvio.TipoDestino;
    TablaActividad.NumeroDestino= LineasEnvio. NumeroDestino;
    TablaActividad.EnvioAsesoramiento = LineasEnvio. EnvioAsesoramiento;
    TablaActividad.TipoDocumento= TablaActividad.TipoDocumento::Envío;
    TablaActividad.DocumentoEnvio= LineasEnvio.Numero;
    TablaActividad.NumeroLineaDocumentoEnvio= LineasEnvio.NumeroLinea;
    TablaActividad.NumeroSubLineaDocumentoEnvio=TablaReparto.Linea Reparto;

    Si TablaReparto."Nº Bulto" <> " entonces
       TablaActividad.No Bulto= TablaReparto.Nº Bulto;
       TablaActividad.Destino Linea = LineasEnvio.Destino Linea::Bulto;
    Sino
       TablaActividad.Destino Linea= LineasEnvio.Destino Linea::Ubicación;

    TablaContenido se inicializa;
    TablaContenido se filtra Codigo Almacen por TablaReparto.Almacen Origen;
    TablaContenido se filtra por CodigoUbicacion por TablaReparto.Ubicacion Origen;
    TablaContenido se posiciona en el primer registro;

    TablaActividad.CodigoZona= TablaContenido.CodigoZona;
    TablaActividad.CodigoUbicación= TablaContenido Codigo;
    TablaActividad.RankingUbicacion= TablaContenido. RankingUbicacion;
    TablaActividad.TipoCodigoUbicacion= TablaContenido. TipoCodigoUbicacion;
    TablaActividad.Sector = TablaContenido.Sector;

    TablaActividad.VALIDA(TablaActividad.CantidadBase,TablaReparto.Cantidad    (base));
    TablaActividad.VALIDA(TablaActividad.CantidadPendienteBase,TablaReparto.
CantidadPendienteBase);
    TablaActividad.VALIDA (TablaActividad.CantidadaManipular, 0);
    TablaActividad.VALIDA (TablaActividad. CantidadaManipularBase, 0);
    TablaActividad.VALIDA (TablaActividad. CantidadManipuladaBase, 0);
    TablaActividad."FechaHora Creacion" = CURRENTDATETIME;

    Si EsDirectoExpedicion(LineasEnvio) entonces
        TablaActividad."Directo Expedicion"=TRUE;
    Se inserta el registro

    numLinea = numLinea + 10000;

    TablaActividad se inicializa;
    TablaActividad.Tipo Actividad = TablaCabActividad.Tipo;
    TablaActividad.Numero= TablaCabActividad.Numero;
    TablaActividad = numLinea;
    TablaActividad.Tipo Accion= TablaActividad.Tipo Accion ::Place;
    TablaActividad.Tipo Origen= LineasEnvio.Tipo Origen;
    TablaActividad.Subtipo Origen= LineasEnvio. Subtipo Origen;
    TablaActividad.NumeroOrigen= LineasEnvio.NumeroOrigen;
    TablaActividad.NumeroLineaOrigen= LineasEnvio. NumeroLineaOrigen;
    TablaActividad. NumeroSubLineaOrigen = TablaReparto.NumeroLineaReparto;
    TablaActividad.DocumentoOrigen= LineasEnvio.DocumentoOrigen;
    TablaActividad.CodigoAlmacen= LineasEnvio.CodigoAlmacen;
    TablaActividad.NumeroEstante= LineasEnvio.NumeroEstante;
    TablaActividad.NumeroProducto=  LineasEnvio. NumeroProducto;
    TablaActividad.CodigoVariante= LineasEnvio. CodigoVariante;
```

```
   TablaActividad.UnidadMedida= LineasEnvio. UnidadMedida;
   TablaActividad.CantidadUnidadMedida= LineasEnvio. CantidadUnidadMedida;
   TablaActividad.Descripcion = LineasEnvio. Descripcion;
   TablaActividad. Descripcion2 = LineasEnvio. Descripcion2;
   TablaActividad.FechaVencimiento= LineasEnvio.FechaVencimiento;
   TablaActividad.FechaInicio= LineasEnvio.FechaEnvio;
   TablaActividad.TipoDestino= LineasEnvio.TipoDestino;
   TablaActividad.NumeroDestino= LineasEnvio. NumeroDestino;
   TablaActividad.EnvioAsesoramiento = LineasEnvio. EnvioAsesoramiento;
   TablaActividad.TipoDocumento= TablaActividad.TipoDocumento::Envío;
   TablaActividad.DocumentoEnvio= LineasEnvio.Numero;
   TablaActividad.NumeroLineaDocumentoEnvio= LineasEnvio.NumeroLinea;
   TablaActividad.NumeroSubLineaDocumentoEnvio=TablaReparto.Linea Reparto;

   Si TablaReparto."Nº Bulto" <> " entonces
      TablaActividad.No Bulto= TablaReparto.Nº Bulto;
      TablaActividad.Destino Linea = LineasEnvio.Destino Linea::Bulto;
   Sino
      TablaActividad.Destino Linea= LineasEnvio.Destino Linea::Ubicación;

   TablaContenido se inicializa;
   TablaContenido se filtra CodigoAlmacen por LineasEnvio.CodigoAlmacen;
   TablaContenido se filtra CodigoUbicacion por LineasEnvio.CodigoUbicacion;
   Se posiciona primer registro TablaContenido

   TablaActividad.CodigoZona= TablaContenido.CodigoZona;
   TablaActividad.CodigoUbicación= TablaContenido Codigo;
   TablaActividad.RankingUbicacion= TablaContenido. RankingUbicacion;
   TablaActividad.TipoCodigoUbicacion= TablaContenido. TipoCodigoUbicacion;
   TablaActividad.Sector = TablaContenido.Sector;

   TablaActividad.VALIDA(TablaActividad.CantidadBase,TablaReparto.Cantidad    (base));
   TablaActividad.VALIDA(TablaActividad.CantidadPendienteBase,TablaReparto.
CantidadPendienteBase);
   TablaActividad.VALIDA (TablaActividad.CantidadaManipular, 0);
   TablaActividad.VALIDA (TablaActividad. CantidadaManipularBase, 0);
   TablaActividad.VALIDA (TablaActividad. CantidadManipuladaBase, 0);
   TablaActividad."FechaHora Creacion" = CURRENTDATETIME;

   Si EsDirectoExpedicion(LineasEnvio) entonces
       TablaActividad."Directo Expedicion"=TRUE;
   Se inserta el registro

   numLinea = numLinea + 10000;
   TablaReparto2 se inicializa;
   TablaReparto2 se filtra NumeroEnvio por TablaReparto.NumeroEnvio;
   TablaReparto2 se filtra NumeroLinEnvio por TablaReparto.NumeroLinEnvio;
   TablaReparto2 se filtra NumeroLinReparto por TablaReparto.NumeroLinReparto;
   Si se encuentra registros TablaReparto2 entonces
     TablaReparto2.Estado = TablaReparto2::"Documento Picking";
     Registro TablaReparto2 se modifica;
UNTIL Reparto.NEXT = 0;

DarPrioridadPicking(TablaCabActividad.Numero);
```

Código Original

```
WhseActivHeader.RESET;
WhseActivHeader.SETRANGE(WhseActivHeader.Type, WhseActivHeader.Type::Pick);
WhseActivHeader.SETRANGE(WhseActivHeader."ExternalDocumentNo.",
NumDocAgrupado);
IF NOT WhseActivHeader.FINDFIRST THEN
BEGIN
  WarehouseSetup.RESET;
  WarehouseSetup.GET;

  WhseActivHeader.RESET;
  WhseActivHeader.INIT;
  WhseActivHeader.Type := WhseActivHeader.Type::Pick;
  WhseActivHeader."Location Code" := InforEmpresa."Location Code";
  WhseActivHeader."Sorting Method" := WhseActivHeader."Sorting Method"::"Action Type";
  WhseActivHeader."External Document No." := PARDocAgrupado;
  WhseActivHeader."Registering No. Series" := WarehouseSetup."Registered Whse. Pick Nos.";
  WhseActivHeader.INSERT(TRUE);
END;

WhseActivLine.RESET;
WhseActivLine.SETRANGE(WhseActivLine."ActivityType",          WhseActivLine."Activity
Type"::Pick);
WhseActivLine.SETRANGE(WhseActivLine."No.", WhseActivHeader."No.");
IF WhseActivLine.FINDLAST THEN
   numLinea := WhseActivLine."Line No." + 10000;

Reparto.RESET;
Reparto.SETRANGE("No Documento Agrupado",PARDocAgrupado);
Reparto.SETRANGE(Reparto.Estado, Reparto.Estado::" ");
Reparto.SETCURRENTKEY("AlmacenOrigen","ZonaOrigen","UbicacionOrigen","
Cod Ud Medida","Cod Producto",Estado);


IF Reparto.FINDSET THEN
REPEAT
   EnvioLinea.RESET;
   EnvioLinea.SETRANGE(EnvioLinea."No.",Reparto."No Envio");
   EnvioLinea.SETRANGE(EnvioLinea."Line No.",Reparto."No Linea Envio");
   EnvioLinea.FINDFIRST;

   WhseActivLine.RESET;
   WhseActivLine.INIT;
   WhseActivLine."Activity Type" := WhseActivHeader.Type;
   WhseActivLine."No." := WhseActivHeader."No.";
   WhseActivLine."Line No." := numLinea;
   WhseActivLine."Action Type" := WhseActivLine."Action Type"::Take;
   WhseActivLine."Source Type" := EnvioLinea."Source Type";
   WhseActivLine."Source Subtype" := EnvioLinea."Source Subtype";
   WhseActivLine."Source No." := EnvioLinea."Source No.";
   WhseActivLine."Source Line No." := EnvioLinea."Source Line No.";
   WhseActivLine."Source Subline No." := Reparto."No Linea Reparto";
```

```
    WhseActivLine."Source Document" := EnvioLinea."Source Document";
    WhseActivLine."Location Code" := EnvioLinea."Location Code";
    WhseActivLine."Shelf No." := EnvioLinea."Shelf No.";
    WhseActivLine."Item No." :=  EnvioLinea."Item No.";
    WhseActivLine."Variant Code" := EnvioLinea."Variant Code";
    WhseActivLine."Unit of Measure Code" := EnvioLinea."Unit of Measure Code";
    WhseActivLine."Qty. per Unit of Measure" := EnvioLinea."Qty. per Unit of Measure";
    WhseActivLine.Description := EnvioLinea.Description;
    WhseActivLine."Description 2" := EnvioLinea."Description 2";
    WhseActivLine."Due Date" := EnvioLinea."Due Date";
    WhseActivLine."Starting Date" := EnvioLinea."Shipment Date";
    WhseActivLine."Destination Type" := EnvioLinea."Destination Type";
    WhseActivLine."Destination No." := EnvioLinea."Destination No.";
    WhseActivLine."Shipping Advice" := EnvioLinea."Shipping Advice";

    WhseActivLine."Whse. Document Type" := WhseActivLine."Whse. Document
Type"::Shipment;
    WhseActivLine."Whse. Document No." := EnvioLinea."No.";
    WhseActivLine."Whse. Document Line No." := EnvioLinea."Line No.";
    WhseActivLine."Whse. Document Sub Line No." := Reparto."No Linea Reparto";

    IF (Reparto."Nº Bulto" <> '') THEN
    BEGIN
      WhseActivLine."No Bulto" := Reparto."Nº Bulto";
      WhseActivLine."Destino Linea" := WhseActivLine."Destino Linea"::Bulto;
    END ELSE BEGIN
      WhseActivLine."Destino Linea" := WhseActivLine."Destino Linea"::Ubicación;
    END;

    Bin.RESET;
    Bin.SETRANGE(Bin."Location Code",Reparto."Almacen Origen");
    Bin.SETRANGE(Bin.Code,Reparto."Ubicacion Origen");
    Bin.FINDFIRST;

    WhseActivLine."Zone Code" := Bin."Zone Code";
    WhseActivLine."Bin Code" := Bin.Code;
    WhseActivLine."Bin Ranking" := Bin."Bin Ranking";
    WhseActivLine."Bin Type Code" := Bin."Bin Type Code";
    WhseActivLine.Sector := Bin.Sector;

    WhseActivLine.VALIDATE(WhseActivLine."Qty.(Base)",Reparto."Cantidad (base)");
    WhseActivLine.VALIDATE(WhseActivLine."Qty.Outstanding(Base)",     Reparto."Cantidad
(base)");
    WhseActivLine.VALIDATE(WhseActivLine."Qty. to Handle (Base)", 0);
    WhseActivLine.VALIDATE(WhseActivLine."Qty. to Handle", 0);
    WhseActivLine.VALIDATE(WhseActivLine."Qty. Handled (Base)", 0);
    WhseActivLine."FechaHora Creacion" := CURRENTDATETIME;

    IF EsDirectoExpedicion(EnvioLinea) THEN
      WhseActivLine."Directo Expedicion":=TRUE;
    WhseActivLine.INSERT(TRUE);

    numLinea := numLinea + 10000;
```

```
  WhseActivLine.RESET;
  WhseActivLine.INIT;
  WhseActivLine."Activity Type" := WhseActivHeader.Type;
  WhseActivLine."No." := WhseActivHeader."No.";
  WhseActivLine."Line No." := numLinea;
  WhseActivLine."Action Type" := WhseActivLine."Action Type"::Place;
  WhseActivLine."Source Type" := EnvioLinea."Source Type";
  WhseActivLine."Source Subtype" := EnvioLinea."Source Subtype";
  WhseActivLine."Source No." := EnvioLinea."Source No.";
  WhseActivLine."Source Line No." := EnvioLinea."Source Line No.";
  WhseActivLine."Source Subline No." := Reparto."No Linea Reparto";
  WhseActivLine."Source Document" := EnvioLinea."Source Document";
  WhseActivLine."Location Code" := EnvioLinea."Location Code";
  WhseActivLine."Shelf No." := EnvioLinea."Shelf No.";
  WhseActivLine."Item No." :=  EnvioLinea."Item No.";
  WhseActivLine."Variant Code" := EnvioLinea."Variant Code";
  WhseActivLine."Unit of Measure Code" := EnvioLinea."Unit of Measure Code";
  WhseActivLine."Qty. per Unit of Measure" := EnvioLinea."Qty. per Unit of Measure";
  WhseActivLine.Description := EnvioLinea.Description;
  WhseActivLine."Description 2" := EnvioLinea."Description 2";
  WhseActivLine."Due Date" := EnvioLinea."Due Date";
  WhseActivLine."Starting Date" := EnvioLinea."Shipment Date";
  WhseActivLine."Destination Type" := EnvioLinea."Destination Type";
  WhseActivLine."Destination No." := EnvioLinea."Destination No.";
  WhseActivLine."Shipping Advice" := EnvioLinea."Shipping Advice";

  WhseActivLine."Whse. Document Type" := WhseActivLine."Whse. Document
Type"::Shipment;
  WhseActivLine."Whse. Document No." := EnvioLinea."No.";
  WhseActivLine."Whse. Document Line No." := EnvioLinea."Line No.";
  WhseActivLine."Whse. Document Sub Line No." := Reparto."No Linea Reparto";

  IF (Reparto."Nº Bulto" <> '') THEN
  BEGIN
    WhseActivLine."No Bulto" := Reparto."Nº Bulto";
    WhseActivLine."Destino Linea" := WhseActivLine."Destino Linea"::Bulto;
  END ELSE BEGIN
    WhseActivLine."Destino Linea" := WhseActivLine."Destino Linea"::Ubicación;
  END;

  Bin.RESET;
  Bin.SETRANGE(Bin."Location Code",EnvioLinea."Location Code");
  Bin.SETRANGE(Bin.Code,EnvioLinea."Bin Code");
  Bin.FINDFIRST;

  WhseActivLine."Zone Code" := Bin."Zone Code";
  WhseActivLine."Bin Code" := Bin.Code;
  WhseActivLine."Bin Ranking" := Bin."Bin Ranking";
  WhseActivLine."Bin Type Code" := Bin."Bin Type Code";
  WhseActivLine.Sector := Bin.Sector;

  WhseActivLine.VALIDATE(WhseActivLine."Qty.(Base)",Reparto."Cantidad (base)");
  WhseActivLine.VALIDATE(WhseActivLine."Qty.Outstanding(Base)",    Reparto."Cantidad
(base)");
```

```
    WhseActivLine.VALIDATE(WhseActivLine."Qty. to Handle (Base)", 0);
    WhseActivLine.VALIDATE(WhseActivLine."Qty. to Handle", 0);
    WhseActivLine.VALIDATE(WhseActivLine."Qty. Handled (Base)", 0);
    WhseActivLine."FechaHora Creacion" := CURRENTDATETIME;

    IF EsDirectoExpedicion(EnvioLinea) THEN
      WhseActivLine."Directo Expedicion":=TRUE;

    WhseActivLine.INSERT(TRUE);

    numLinea := numLinea + 10000;

    RepartoMod.RESET;
    RepartoMod.SETRANGE(RepartoMod."No Envio",Reparto."No Envio");
    RepartoMod.SETRANGE(RepartoMod."No Linea Envio",Reparto."No Linea Envio");
    RepartoMod.SETRANGE(RepartoMod."No Linea Reparto",Reparto."No Linea Reparto");
    IF RepartoMod.FINDSET(TRUE) THEN
    BEGIN
      RepartoMod.Estado := Reparto.Estado::"Documento Picking";
      RepartoMod.MODIFY;
    END;
UNTIL Reparto.NEXT = 0;

DarPrioridadPicking(WhseActivHeader."No.");
```

EsDirectoExpedicion

PseudoCódigo

```
Si LineasEnvios.CodigoZona=’EXPEDICION’ entonces
    Devuelve Verdadero
Sino
    Devuelve Falso
```

Código Original

```
IF PAREnvio."Zone Code" = 'EXPEDICION' THEN
   EXIT(TRUE)
ELSE
  EXIT(FALSE);
```

DarPrioridadPicking

PseudoCódigo

```
TablaActividad se inicializa
TablaActividad se filtra Tipo Actividad sea Picking
TablaActividad se filtra Trilateral sea Falso
Si encuentra ultimo registro TablaActividad entonces
 Si no evalua nvPrioridad con TablaActividad.Prioridad entonces
     nvPrioridad:=1000000;

TablaActividad se inicializa;
TablaActividad se filtra Tipo Actividad sea Picking;
TablaActividad se filtra Número por PARPicking;
```

TablaActividad se filtra Tipo Accion por Traer;
TablaActividad se filtra Trilateral sea Falso;

Si encuentra registros TablaActividad entonces
REPEAT
nvPrioridad+=1;
cvPrioridad:=FORMAT(nvPrioridad);
TablaActividad.Prioridad:=cvPrioridad;
Se modifica registro TablaActividad

TablaActividad2 se inicializa
TablaActividad2 se filtra Tipo Origen por TablaActividad.TipoOrigen;
TablaActividad2 se filtra Subtipo Origen por TablaActividad.SubtipoOrigen;
TablaActividad2 se filtra Numero Origen por TablaActividad.NumeroOrigen;
TablaActividad2 se filtra Numero Linea Origen por TablaActividad.NumLinOrigen;
TablaActividad2 se filtra Numero SubLinea Origen por TablaActividad.NumSubLinOrigen;
TablaActividad2 se filtra Numero Bulto por TablaActividad.NumBulto; TablaActividad2 se filtra Tipo Documento por TablaActividad.TipoDocumento;
TablaActividad2 se filtra Numero Documento por TablaActividad.NumDocumento;
TablaActividad2 se filtra Numero Linea Documento por TablaActividad.NumLinDocumento;
TablaActividad2 se filtra Numero SubLinea Documento por TablaActividad.NumSubLinDocumento;
TablaActividad2 se filtra por tipo accion Place
TablaActividad2 se filtra número por TablaActividad.Num
TablaActividad2 se filtra TipoActividad por TablaActividad.TipoActividad;

Si no encuentro registros TablaActividad2 entonces
ERROR(‘No existe linea’);
TablaActividad2.Prioridad:=cvPrioridad;
Se modifica registro TablaActividad2;
Hasta que siguiente registro TablaActividad sea vacío

Código Original

```
rcdPicking.RESET;
rcdPicking.SETCURRENTKEY("Activity Type",Prioridad,"No.","Line No.","Action Type","Bin Code");
rcdPicking.SETRANGE("Activity Type",rcdPicking."Activity Type"::Pick);
rcdPicking.SETRANGE(Trilateral, FALSE);
IF rcdPicking.FINDLAST THEN
  IF NOT EVALUATE(nvPrioridad,rcdPicking.Prioridad) THEN
    nvPrioridad:=1000000;

rcdPicking.RESET;
rcdPicking.SETCURRENTKEY("Activity Type","No.","Bin Code","Breakbulk No.","Action Type");
rcdPicking.SETRANGE("Activity Type",rcdPicking."Activity Type"::Pick);
rcdPicking.SETRANGE("No.",PARPicking);
rcdPicking.SETRANGE("Action Type",rcdPicking."Action Type"::Take);
.rcdPicking.SETRANGE(Trilateral, FALSE);
IF rcdPicking.FINDSET THEN
REPEAT
 nvPrioridad+=1;
 cvPrioridad:=FORMAT(nvPrioridad);
```

```
  rcdPicking.Prioridad:=cvPrioridad;
  rcdPicking.MODIFY;

  Lin2.RESET;
  Lin2.SETCURRENTKEY("Source Type","Source Subtype","Source No.","Source Line
No.","Source Subline No.",
                "Unit of Measure Code","Action Type","Breakbulk No.","Original Breakbulk");
  Lin2.SETRANGE("Source Type",rcdPicking."Source Type");
  Lin2.SETRANGE("Source Subtype",rcdPicking."Source Subtype");
  Lin2.SETRANGE("Source No.",rcdPicking."Source No.");
  Lin2.SETRANGE("Source Line No.",rcdPicking."Source Line No.");
  Lin2.SETRANGE("Source Subline No.",rcdPicking."Source Subline No.");
  Lin2.SETRANGE("No Bulto",rcdPicking."No Bulto");
  Lin2.SETRANGE("Whse. Document Type", rcdPicking."Whse. Document Type");
  Lin2.SETRANGE("Whse. Document No.", rcdPicking."Whse. Document No.");
  Lin2.SETRANGE("Whse. Document Line No.", rcdPicking."Whse. Document Line No.");
  Lin2.SETRANGE("Whse. Document Sub Line No.",rcdPicking."Whse. Document Sub Line
No.");
  Lin2.SETRANGE("Action Type",Lin2."Action Type"::Place);
  Lin2.SETRANGE("No.",rcdPicking."No.");
  Lin2.SETRANGE("Activity Type",rcdPicking."Activity Type");
  IF NOT Lin2.FINDFIRST THEN
    ERROR('No existe el colocar para %1
%2',FORMAT(rcdPicking."No."),FORMAT(rcdPicking."Line No."));
  Lin2.Prioridad:=cvPrioridad;
  Lin2.MODIFY;
UNTIL rcdPicking.NEXT=0;
```

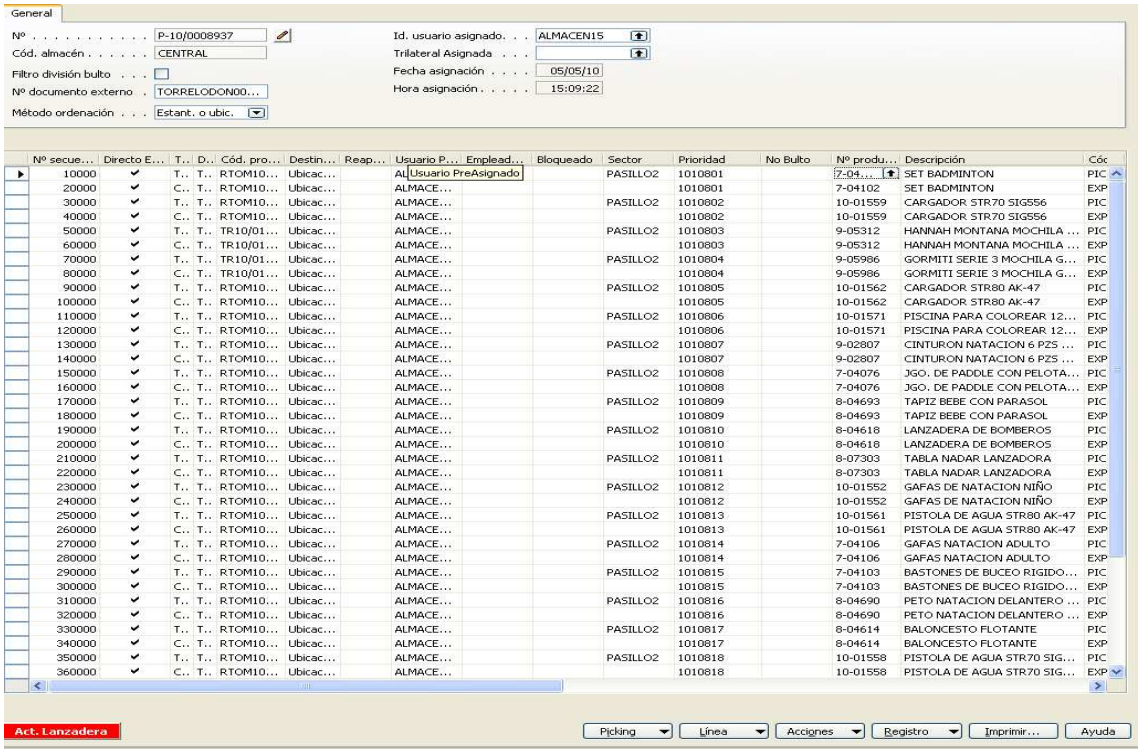


Ilustración 28 Hoja de Picking

Reaprovisionamiento

Este apartado consiste en dos partes bien definidas, que son las siguientes:

Buscar Reserva a Bajar

En este caso, el sistema se recorre todas las líneas de envíos existentes. Para cada línea realiza lo siguiente:

En una tabla temporal comprueba si el producto correspondiente a la línea de envío se encuentra en la tabla.

En caso de no existir, se inserta el producto en la tabla. Pero antes de la inserción, se debe calcular la cantidad en picking que existe del producto, en el campo cantidad envío se debe añadir la cantidad que indica la línea del envío y calcular la cantidad necesaria, que en este caso es la resta de la cantidad de envío menos la cantidad en picking. Una vez calculados los campos, es el momento de realizar la inserción del registro en la tabla.

En caso de existir el registro, lo que se debe hacer es actualizar los campos cantidad envío y cantidad necesaria con los datos de la línea del envío.

PseudoCódigo

```
TablaTemporal se inicializa
TablaTemporal se borran todos los registros existentes
TablaLinEnvios se inicializa
Si existen registros TablaLinEnvios entonces
    Repetir
      Si HayQueProcesarEnvio(TablaLinEnvios) entonces
        TablaTemporal se inicializa
        TablaTemporal se filtra Producto por TablaLinEnvio.NumeroProducto
         Si existe Registro TablaTemporal entonces
            TablaTemporal.CantidadEnvio+=TablaLinEnvio.Cantidad-
           TablaLinEnvio.CantidadPreparada-TablaLinEnvio.CantidadEnviada
            TablaTemporal.CantidadNecesaria:=TablaTemporal.CantidadEnvio-
           TablaTemporal.CantidadPicking
            Registro TablaTemporal se modifica
      Sino
           TablaTemporal se inicializa
            TablaTemporal.CantidadEnvio+=TablaLinEnvio.Cantidad-
           TablaLinEnvio.CantidadPreparada-TablaLinEnvio.CantidadEnviada
            TablaProducto se inicializa
            TablaProducto se filtra Número por TablaLinEnvio.NumeroProducto
            TablaProducto se filtra Filtro Ubicación por sea distinto ISLA y RTOMASIVO
            Si encuentra Registro TablaProducto entonces
                Se calcula Existencias Picking en TablaProducto
                TablaTemporal:=TablaProducto.ExistenciasPicking
            TablaTemporal.CantidadNecesaria:=TablaTemporal.CantidadEnvio-
           TablaTemporal.CantidadPicking
    Hasta que Registro TablaLinEnvios sea vacío
```

Código Original

```
Temporal.RESET;
Temporal.DELETEALL;
ShipmentL.RESET;
ShipmentL.SETCURRENTKEY("Item No.","Location Code","Variant Code","Due Date");
IF ShipmentL.FINDSET(FALSE, FALSE) THEN
BEGIN
  REPEAT
   IF (FuncionesIñaki.HayQueProcesarEnvio(ShipmentL)) THEN
   BEGIN
     Temporal.RESET;
     Temporal.SETRANGE(Temporal.Producto,  ShipmentL."Item No.");
     IF (Temporal.FINDFIRST) THEN
     BEGIN
       Temporal."Cantidad Envio" +=  ShipmentL.Quantity - ShipmentL."Qty. Picked" -
ShipmentL."Qty. Shipped";
       Temporal."Cantidad Necesaria" := Temporal."Cantidad Envio" - Temporal."Cantidad
Picking";
       Temporal.MODIFY;
     END ELSE BEGIN
       Temporal.INIT;
       Temporal.Producto :=  ShipmentL."Item No.";
       Temporal."Cantidad Envio" := ShipmentL.Quantity - ShipmentL."Qty. Picked" -
ShipmentL."Qty. Shipped";
       Item.RESET;
       Item.SETRANGE("No.", ShipmentL."Item No.")
       Item.SETFILTER("Bin Filter", '<>ISLA&<>RTOMASIVO');
       IF Item.FINDFIRST THEN
       BEGIN
         Item.CALCFIELDS(Item."Existencias Picking");
         Temporal."Cantidad Picking" := Item."Existencias Picking";
       END;

       Temporal."CantidadNecesaria”:=Temporal."CantidadEnvio"-  Temporal."Cantidad
Picking";
       Temporal.INSERT;
     END;
   END;
  UNTIL ShipmentL.NEXT = 0;
END;
```

HayQueProcesarEnvio

PseudoCódigo

```
Case TablaLinEnvio.Documento Origen Of
      TablaLinEnvio.Documento Origen es igual Pedido Venta
            TablaPlanificacion se inicializa
             TablaPlanificacion se filtra Tipo Documento sea Pedido Venta
            TablaPlanificacion se filtra Transferir a por TablaLinEnvio. NumeroOrigen
            TablaPlanificacion se filtra Habilitada sea Verdadero
            Si encuentra registro TablaPlanificacion entonces
                  Devolver Verdadero
             TablaPlanificacion se inicializa
```

```
            TablaPlanificacion se filtra Tipo Documento sea Cliente
            TablaPlanificacion se filtra Transferir a por TablaLinEnvio. NumeroDestino
            TablaPlanificacion se filtra Habilitada sea Verdadero
            Si encuentra registro TablaPlanificacion entonces
                Devolver Verdadero
    TablaLinEnvio.Documento Origen es igual Transferencia
            TablaPlanificacion se inicializa
            TablaPlanificacion se filtra Tipo Documento sea Transferencia
            TablaPlanificacion se filtra Transferir a por TablaLinEnvio. NumeroDestino
            TablaPlanificacion se filtra Habilitada sea Verdadero
            Si encuentra registro TablaPlanificacion entonces
                Devolver Verdadero
Devolver Falso
```

Código Original

```
CASE ShipmentL."Source Document" OF
    ShipmentL."Source Document"::"Sales Order":
      BEGIN
        rcdPlanif.RESET;
        rcdPlanif.SETRANGE("Tipo Documento",rcdPlanif."Tipo Documento"::"Pedido
Venta");
        rcdPlanif.SETRANGE("Transferir a",ShipmentL."Source No.");
        rcdPlanif.SETRANGE(rcdPlanif.Habilitada,TRUE);
        IF rcdPlanif.FINDFIRST THEN
          EXIT(TRUE);

        rcdPlanif.RESET;
        rcdPlanif.SETRANGE("Tipo Documento",rcdPlanif."Tipo Documento"::Cliente);
        rcdPlanif.SETRANGE("Transferir a",ShipmentL."Destination No.");
        rcdPlanif.SETRANGE(rcdPlanif.Habilitada,TRUE);
        IF rcdPlanif.FINDFIRST THEN
          EXIT(TRUE);
      END;
    ShipmentL."Source Document"::"Outbound Transfer":
      BEGIN
        rcdPlanif.RESET;

rcdPlanif.SETRANGE("TipoDocumento",rcdPlanif."TipoDocumento"::Transferencia);
        rcdPlanif.SETRANGE("Transferir a",ShipmentL."Destination No.");
        rcdPlanif.SETRANGE(rcdPlanif.Habilitada,TRUE);
        IF rcdPlanif.FINDFIRST THEN
          EXIT(TRUE);
      END;
END;
EXIT(FALSE);
```

Generar Orden de Bajada

En esta función, el sistema hace lo siguiente:

Se recorre cada uno de los registros de la tabla temporal. Para cada registro hace lo siguiente:

En primer lugar, comprueba que la cantidad servida sea menor que cantidad necesaria. En caso de que se cumpla, el sistema se recorre todos los palets que

contengan el producto que marca el registro de la tabla temporal. Para cada palet se realiza una serie de comprobaciones:

Comprueba que el palet contiene una sola referencia y comprueba que esté en reserva y cerrado. En caso de cumplir esos requisitos, el sistema da orden de bajada del palet y genera la hoja de picking con las órdenes correspondientes. Una vez generadas esas órdenes, se debe modificar la cantidad servida y la cantidad localizada. A continuación, se comprueba si cantidad localizada es mayor que cantidad necesaria. En caso de que se cumpla la premisa, el sistema deja de buscar palets para reaprovisionar porque la demanda de pedidos ha sido copada. Si no se cumple la premisa, se siguen buscando palets para reaprovisionar.

En caso de que el palet sea de varias referencias, se realiza el mismo proceso que para los de una sola referencia.

PseudoCódigo

```
TablaTemporal se inicializa
Si encuentra registros TablaTemporal entonces
Repetir
    Si TablaTemporal.CantidadServida < TablaTemporal.CantidadNecesaria
        TablaBultos se inicializa
        TablaBultos se filtra por Producto sea TablaTemporal.Producto
        Si encuentra registros TablaBultos entonces
          Repetir
            TablaBultos2 se inicializa
              TablaBultos2 se filtra Bulto sea igual TablaBultos.Bulto
              Si TablaBultos2.NumeroLineas=1 entonces
                 TablaBultos3 se inicializa
                 TablaBultos3 se filtra Bulto sea igual TablaBultos.Bulto
                 Si encuentra registros TablaBultos3 entonces
                  Si TablaBultos3.TipoBulto=Reserva y TablaBultos3.ZonaActual=
                      Reserva y TablaBultos3.Estado=Cerrado y
                      TablaBultos2.ZonaDestino='' y TablaBultos3.EstadoDestino=''
                      entonces
                        TablaBultos3.ZonaDestino='PICKING';
                        TablaBultos3.EstadoDestino='ISLA';
                       TablaBultos3.DocaRegistrar='DocMovimiento'
                        Modificar Registro TablaBultos3
                         CrearDocMovBajarBulto(TablaBultos3.No Bulto)
                         ReaprovisionarPicking(TablaBultos3.No Bulto)
                       TablaTemporal2 se inicializa
                          TablaTemporal2 se filtra Producto por
                       TablaTemporal.Producto
                          Si encuentra registro TablaTemporal2 entonces
                          TablaTemporal2.CantidadServida=
                              TablaTemporal.CantidadServida
                       TablaBultos3.EstadoDestino='Reaprovisionamiento'
```

```
                                Modificar Registro TablaBultos3
                                  TablaBultos4 se inicializa
                                  TablaBultos4 se filtra Producto sea distinto
                                  Si encuentra registros TablaBultos4 entonces
                                 Repetir
                                  TablaTemporal2 se inicializa
                                     TablaTemporal2 se filtra Producto sea
                                     TablaBultos4.Producto
                                     Si encuentra registro TablaTemporal2 entonces
                                        TablaTemporal2.CantidadLocalizada=
                                        TablaBultos4.Cantidad
                                  Hasta registro TablaBultos4 sea vacío
                                  TablaTemporal.CantidadLocalizada+=
                                  TablaBultos.Cantidad
                                  Modifica Registro TablaTemporal
                                  Si TablaTemporal.CantidadLocalizada>=
                                  TablaTemporal.CantidadNecesaria entonces
                                        salir=verdadero;
            Hasta registro TablaBultos sea vacío o salir sea verdadero
 Sino encuentra registros TablaBultos entonces
       noseguir=verdadero;
Si noseguir=falso entonces
   Si salir=falso entonces
     TablaBultos se inicializa
     TablaBultos se filtra por Producto sea TablaTemporal.Producto
     Si encuentra registros TablaBultos entonces
     Repetir
        TablaBultos3 se inicializa
        TablaBultos3 se filtra Bulto sea igual TablaBultos.Bulto
        Si encuentra registros TablaBultos3 entonces
           Si TablaBultos3.TipoBulto=Reserva y TablaBultos3.ZonaActual=
              Reserva y TablaBultos3.Estado=Cerrado y
              TablaBultos2.ZonaDestino='' y TablaBultos3.EstadoDestino=''
               entonces
                 TablaBultos3.ZonaDestino='PICKING';
                 TablaBultos3.EstadoDestino='ISLA';
                TablaBultos3.DocaRegistrar='DocMovimiento'
                  Modificar Registro TablaBultos3
                 CrearDocMovBajarBulto(TablaBultos3.No Bulto)
                 ReaprovisionarPicking(TablaBultos3.No Bulto)
                TablaTemporal2 se inicializa
                 TablaTemporal2 se filtra Producto por
                 TablaTemporal.Producto
                   Si encuentra registro TablaTemporal2 entonces
                     TablaTemporal2.CantidadServida=
                       TablaTemporal.CantidadServida
                    TablaBultos3.EstadoDestino='Reaprovisionamiento'
                    Modificar Registro TablaBultos3
                       TablaBultos4 se inicializa
                       TablaBultos4 se filtra Producto sea distinto
                       Si encuentra registros TablaBultos4 entonces
                     Repetir
                       TablaTemporal2 se inicializa
                         TablaTemporal2 se filtra Producto sea
```

```
                                    TablaBultos4.Producto
                                     Si encuentra registro TablaTemporal2 entonces
                                         TablaTemporal2.CantidadLocalizada=
                                         TablaBultos4.Cantidad
                                   Hasta registro TablaBultos4 sea vacío
                                   TablaTemporal.CantidadLocalizada+=
                                   TablaBultos.Cantidad
                                   Modifica Registro TablaTemporal
                                   Si TablaTemporal.CantidadLocalizada>=
                                   TablaTemporal.CantidadNecesaria entonces
                                      salir=verdadero;
                          Hasta registro TablaBultos sea vacío o salir sea verdadero
Hasta registro TablaTemporal sea vacío
```

Código Original

```
Temporal.RESET;
IF Temporal.FIND('-') THEN
REPEAT
   IF (Temporal."Cantidad Servida" < Temporal."Cantidad Necesaria") THEN
   BEGIN
      bSalir := FALSE;
      bNoSeguir := FALSE;
      BultosL.RESET;
      BultosL.SETCURRENTKEY(Producto,Cantidad);
      BultosL.ASCENDING(FALSE);
      BultosL.SETRANGE(BultosL.Producto, Temporal.Producto);
      IF BultosL.FIND('-') THEN
      BEGIN
         REPEAT
            BultosLaux.RESET;
            BultosLaux.SETRANGE(BultosLaux."No Bulto", BultosL."No Bulto");
            IF BultosLaux.COUNT = 1 THEN
            BEGIN
               BultosH.RESET;
               BultosH.SETRANGE(BultosH."No Bulto", BultosL."No Bulto");
               IF BultosH.FINDFIRST THEN
               BEGIN
                  IF (BultosH."Tipo Bulto" = BultosH."Tipo Bulto"::Reserva) AND
(BultosH."Zona Actual" = 'RESERVA') AND
                     (BultosH."Estado Actual" = BultosH."Estado Actual"::Cerrado) AND
(BultosH."Zona Destino" = '') AND
                     (BultosH."Estado Destino" = BultosH."Estado Destino"::" ")  THEN
                  BEGIN
                     BultosH."Zona Destino" := 'PICKING';
                     BultosH."Ubicacion Destino" := 'ISLA';
                     BultosH."Estado Destino" := BultosH."Estado Destino"::" ";
                     BultosH."Doc. A Registrar" := BultosH."Doc. A Registrar"::DocMovimiento;
                     BultosH.MODIFY;
                     Funciones.CrearDocMovBajarBulto(BultosH."No Bulto");
                     ReaprovisionarPicking(BultosH."No Bulto");
                     TemporalAux.RESET;
                     TemporalAux.SETRANGE(TemporalAux.Producto, Temporal.Producto);
                     IF TemporalAux.FINDFIRST THEN
```

```
                    Temporal."Cantidad Servida" := TemporalAux."Cantidad Servida";

                BultosH."Estado Destino" := BultosH."Estado Destino"::Reaprovisionamiento;
                BultosH.MODIFY;
                BultosLaux2.RESET;
                BultosLaux2.SETRANGE("No Bulto", BultosH."No Bulto");
                BultosLaux2.SETFILTER(Producto,'<>%1',Temporal.Producto);
                IF BultosLaux2.FINDSET THEN
                REPEAT
                  TemporalAux.RESET;
                  TemporalAux.SETRANGE(TemporalAux.Producto, BultosLaux2.Producto);
                  IF TemporalAux.FINDFIRST THEN
                  BEGIN
                    TemporalAux."Cantidad Localizada" += BultosLaux2.Cantidad;
                    TemporalAux.MODIFY;
                  END;
                UNTIL BultosLaux2.NEXT = 0;
                Temporal."Cantidad Localizada" += BultosL.Cantidad;
                Temporal.MODIFY;
                IF (Temporal."Cantidad Localizada" >= Temporal."Cantidad Necesaria") THEN
                  bSalir := TRUE;

              END
            END;
          END;

      UNTIL (BultosL.NEXT = 0) OR (bSalir);
    END ELSE
    BEGIN
      bNoSeguir := TRUE;
    END;

    IF (NOT bNoSeguir) THEN
    BEGIN

      IF NOT bSalir THEN
      BEGIN
        BultosL.RESET;
        BultosL.SETCURRENTKEY(Producto,Cantidad);
        BultosL.ASCENDING(FALSE);
        BultosL.SETRANGE(BultosL.Producto, Temporal.Producto);
        IF BultosL.FIND('-') THEN
        REPEAT
          BultosH.RESET;
          BultosH.SETRANGE(BultosH."No Bulto", BultosL."No Bulto");
          IF BultosH.FINDFIRST THEN
          BEGIN
            IF (BultosH."Tipo Bulto" = BultosH."Tipo Bulto"::Reserva) AND (BultosH."Zona Actual" = 'RESERVA') AND
              (BultosH."Estado Actual" = BultosH."Estado Actual"::Cerrado) AND (BultosH."Zona Destino" = '') AND
```

```
                (BultosH."Estado Destino" = BultosH."Estado Destino"::" ")  THEN
              BEGIN
                BultosH."Zona Destino" := 'PICKING';
                BultosH."Ubicacion Destino" := 'ISLA';
                BultosH."Estado Destino" := BultosH."Estado Destino"::" ";
                BultosH."Doc. A Registrar" := BultosH."Doc. A Registrar"::DocMovimiento;
                BultosH.MODIFY;
                Funciones.CrearDocMovBajarBulto(BultosH."No Bulto");
                ReaprovisionarPicking(BultosH."No Bulto");
                TemporalAux.RESET;
                TemporalAux.SETRANGE(TemporalAux.Producto, Temporal.Producto);
                IF TemporalAux.FINDFIRST THEN
                  Temporal."Cantidad Servida" := TemporalAux."Cantidad Servida";

                BultosH."Estado Destino" := BultosH."Estado
Destino"::Reaprovisionamiento;
                BultosH.MODIFY;
                BultosLaux.RESET;
                BultosLaux.SETRANGE("No Bulto", BultosH."No Bulto");
                BultosLaux.SETFILTER(Producto,'<>%1',Temporal.Producto);
                IF BultosLaux.FINDSET THEN
                REPEAT
                  TemporalAux.RESET;
                  TemporalAux.SETRANGE(TemporalAux.Producto, BultosLaux.Producto);
                  IF TemporalAux.FINDFIRST THEN
                  BEGIN
                    TemporalAux."Cantidad Localizada" += BultosLaux.Cantidad;
                    TemporalAux.MODIFY;
                  END;
                UNTIL BultosLaux.NEXT = 0;
                Temporal."Cantidad Localizada" += BultosL.Cantidad;
                Temporal.MODIFY;
                IF (Temporal."Cantidad Localizada" >= Temporal."Cantidad Necesaria")
THEN
                  bSalir := TRUE;
              END;
            END;
          UNTIL (BultosL.NEXT = 0) OR (bSalir);
        END;
      END; // FIN DE IF NOT BSEGUIR
  END;
UNTIL Temporal.NEXT = 0;
```

CrearDocMovBajarBulto (sbulto)

PseudoCódigo

```
TablaBultos se inicializa
TablaBultos se filtra NoBulto sea sbulto
Si encuentra registro TablaBultos entonces
   Si TablaBultos.TipoBulto=Reserva y TablaBultos.EstadoDestino<>'' entonces
      ERROR ('Existe un movimiento pendiente');
  Sino Si TablaBultos.TipoBulto=Expedición y TablaBultos.EstadoActual=Cerrado
  entonces
      ERROR(' El bulto ya esta cerrado');
```

ConfigAlmacen se inicializa
ConfigAlmacen se coge el primer registro
TablaCabActividad se inicializa
TablaCabActividad.Tipo=Movimiento
TablaCabActividad.CodigoAlmacen=TablaBultos.Almacen
TablaCabActividad.NoDocumento=sbulto
TablaCabActividad.NoSeriesRegistro=ConfigAlmacen.RegistroMovimientoSerie
Si TablaBultos.ZonaDestino=Reserva
    TablaCabActividad.Accion=Subir
Si TablaBultos.ZonaDestino=Picking
    TablaCabActividad.Accion=Bajar
Inserta registro TablaCabActividad
numLinea=10000;
TablaLinBultos se inicializa
TablaLinBultos se filtra NoBulto sea sbulto
TablaLinBultos se filtra tipo sea producto
Si encuentra registros TablaLinBultos entonces
Repetir
    TablaProducto se inicializa
    TablaProducto se filtra No sea TablaLinBultos.Producto
    TablaProducto se posiciona primer registro
    TablaContenido se inicializa
   TablaContenido se filtra CodigoAlmacen sea TablaBultos.Almacen
   TablaContenido se filtra Codigo sea TablaBultos.Ubicacion Actual
   TablaContenido se posiciona primer registro
   TablaLinActividad se inicializa
   TablaLinActividad.TipoActividad=Movimiento
   TablaLinActividad.No=TablaCabActividad.No
   TablaLinActividad.NoLinea=numLinea
   TablaLinActividad.TipoAccion=Traer
   TablaLinActividad.CodigoAlmacen=TablaBultos.Almacen
   TablaLinActividadCodigoZona=TablaBultos.ZonaActual
   TablaLinActividad.CodigoUbicacion=TablaBultos.UbicacionActual
   TablaLinActividad.NoProducto=TablaLinBultos.Producto
   TablaLinActividad.Descripcion=TablaProducto.Descripcion
   TablaLinActividad.UdMedida=TablaLinBultos.UdMedida
   TablaLinActividad.CantUdMedida=TablaLinBultos.CantUdMedida
   TablaLinActividad.FechaInicio=FechaTrabajo
   TablaLinActividad.RankingUbicacion=TablaContenido.RankingUbicacion
   TablaLinActividad.TipoCódigoUbicacion=TablaContenido. TipoCódigoUbicacion
   TablaLinActividad.Cantidad=TablaLinBultos.Cantidad (base)
   TablaLinActividad.Cantidad (base)=TablaLinBultos.Cantidad (base)
   TablaLinActividad.CantidadManipular=TablaLinBultos.Cantidad (base)
   TablaLinActividad.CantidadManipular (base)=TablaLinBultos.Cantidad (base)
   TablaLinActividad.CantidadPendiente=TablaLinBultos.Cantidad (base)
   TablaLinActividad.CantidadPendiente (base)=TablaLinBultos.Cantidad (base)
   TablaLinActividad.NoOrigen=TablaLinBultos.NoBulto
   TablaLinActividad.NoLineaOrigen=TablaLinBultos.NoLinea
   TablaLinActividad.TipoDocAlmacen=Hoja de Movimiento
   TablaLinActividad.NoDocAlmacen=TablaLinBultos.NoBulto
   TablaLinActividad.NoLinDocAlmacen=TablaLinBultos.NoLinea
   Si sbulto<>’’ entonces
      TablaLinActividad.DestinoLinea=Bulto

TablaLinActividad.NoSubLinDocAlmacen=TablaLinBultos.NoLinea
TablaLinActividad.NoBulto=sbulto
Si TablaBultos.TipoBulto=Reserva entonces
TablaLinActividad.Trilateral=Verdadero
Sino
TablaLinActividad.DestinoLinea=Ubicación
TablaLinActividad.FechaHoraCreacion=Hoy
Insertar Registro TablaLinActividad
Numlinea+=10000;

TablaLinActividad se inicializa
TablaLinActividad.TipoActividad=Movimiento
TablaLinActividad.No=TablaCabActividad.No
TablaLinActividad.NoLinea=numLinea
TablaLinActividad.TipoAccion=Colocar
TablaLinActividad.CodigoAlmacen=TablaBultos.Almacen
TablaLinActividadCodigoZona=TablaBultos.ZonaDestino
TablaLinActividad.CodigoUbicacion=TablaBultos.UbicacionDestino
TablaLinActividad.NoProducto=TablaLinBultos.Producto
TablaLinActividad.Descripcion=TablaProducto.Descripcion
TablaLinActividad.UdMedida=TablaLinBultos.UdMedida
TablaLinActividad.CantUdMedida=TablaLinBultos.CantUdMedida
TablaLinActividad.FechaInicio=FechaTrabajo
TablaLinActividad.RankingUbicacion=TablaContenido.RankingUbicacion
TablaLinActividad.TipoCódigoUbicacion=TablaContenido. TipoCódigoUbicacion
TablaLinActividad.Cantidad=TablaLinBultos.Cantidad (base)
TablaLinActividad.Cantidad (base)=TablaLinBultos.Cantidad (base)
TablaLinActividad.CantidadManipular=TablaLinBultos.Cantidad (base)
TablaLinActividad.CantidadManipular (base)=TablaLinBultos.Cantidad (base)
TablaLinActividad.CantidadPendiente=TablaLinBultos.Cantidad (base)
TablaLinActividad.CantidadPendiente (base)=TablaLinBultos.Cantidad (base)
TablaLinActividad.NoOrigen=TablaLinBultos.NoBulto
TablaLinActividad.NoLineaOrigen=TablaLinBultos.NoLinea
TablaLinActividad.TipoDocAlmacen=Hoja de Movimiento
TablaLinActividad.NoDocAlmacen=TablaLinBultos.NoBulto
TablaLinActividad.NoLinDocAlmacen=TablaLinBultos.NoLinea
Si sbulto<>’’ entonces
TablaLinActividad.DestinoLinea=Bulto
TablaLinActividad.NoSubLinDocAlmacen=TablaLinBultos.NoLinea
TablaLinActividad.NoBulto=sbulto
Si TablaBultos.TipoBulto=Reserva entonces
TablaLinActividad.Trilateral=Verdadero
Sino
TablaLinActividad.DestinoLinea=Ubicación
TablaLinActividad.FechaHoraCreacion=Hoy
Insertar Registro TablaLinActividad
Numlinea+=10000;
TablaContenidoUbicacion se inicializa
TablaContenidoUbicacion se filtra CodigoAlmacen sea TablaBultos.Almacen
TablaContenidoUbicacion se filtra CodigoUbicacion sea
TablaContenido.CodigoUbicacion
TablaContenidoUbicacion se filtra NoProducto sea TablaLinBultos.Producto
TablaContenidoUbicacion se filtra UdMedida sea TablaLinBultos.UdMedida
Si no encuentra registros TablaContenidoUbicacion entonces

```
    TablaContenidoUbicacion se inicializa
    TablaContenidoUbicacion.CodigoAlmacen=TablaBultos.Almacen
    TablaContenidoUbicacion.CodigoUbicacion=TablaContenido.CodigoUbicacion
    TablaContenidoUbicacion.CodigoZona=TablaContenido.CodigoZona
    TablaContenidoUbicacion.NoProducto=TablaLinBultos.Producto
    TablaContenidoUbicacion.NoBulto=sbulto
    Inserta Registro TablaContenidoUbicacion
    TablaContenidoUbicacion.CodigoUbicacion=TablaContenido.CodigoUbicacion
    TablaContenidoUbicacion.CodigoZona=TablaContenido.CodigoZona
    TablaContenidoUbicacion.NoBulto=TablaLinBultos.NoBulto
    Modifica Registro TablaContenidoUbicacion
Hasta registro TablaLinBultos sea vacío
```

Código Original

```
CabeceraBulto.RESET;
CabeceraBulto.SETRANGE("No Bulto",sBulto);
IF CabeceraBulto.FINDFIRST THEN
  IF (CabeceraBulto."Tipo Bulto" = CabeceraBulto."Tipo Bulto"::Reserva)
    AND (CabeceraBulto."Estado Destino" <> CabeceraBulto."Estado Destino"::" ") THEN
  BEGIN
    MESSAGE('EXISTE UN MOVIMIENTO DE ALMACEN.');
    EXIT(FALSE);
  END ELSE IF (CabeceraBulto."Tipo Bulto" = CabeceraBulto."Tipo Bulto"::Expedicion)
        AND (CabeceraBulto."Estado Actual" = CabeceraBulto."Estado Actual"::Cerrado) THEN
  BEGIN
    MESSAGE('EL BULTO YA ESTA CERRADO');
    EXIT(FALSE);
  END;
WarehouseSetup.RESET;
WarehouseSetup.GET;

WhseActivHeader.RESET;
WhseActivHeader.INIT;
WhseActivHeader.Type := WhseActivHeader.Type::Movement;
WhseActivHeader."Location Code" := CabeceraBulto.Almacen;
WhseActivHeader."External Document No." := sBulto;
WhseActivHeader."Registering No. Series" := WarehouseSetup."Registered Whse. Movement
Nos.";

IF (CabeceraBulto."Zona Destino" = 'RESERVA') THEN
   WhseActivHeader.Accion := WhseActivHeader.Accion::Subir
ELSE IF (CabeceraBulto."Zona Destino" = 'PICKING') THEN
   WhseActivHeader.Accion := WhseActivHeader.Accion::Bajar;

WhseActivHeader.INSERT(TRUE);
numLinea := 10000;
LineasBulto.RESET;
LineasBulto.SETRANGE(LineasBulto."No Bulto",sBulto);
LineasBulto.SETRANGE(LineasBulto."Tipo Linea", LineasBulto."Tipo Linea"::Producto);
IF LineasBulto.FINDSET THEN
REPEAT
  Producto.RESET;
  Producto.SETRANGE(Producto."No.",LineasBulto.Producto);
```

```
  Producto.FINDFIRST;
  Bin.RESET;
  Bin.SETRANGE(Bin."Location Code",CabeceraBulto.Almacen);
  Bin.SETRANGE(Bin.Code,CabeceraBulto."Ubicacion Actual");
  Bin.FINDFIRST;

  WhseActivLine.RESET;
  WhseActivLine.INIT;
  WhseActivLine."Activity Type" := WhseActivLine."Activity Type"::Movement;
  WhseActivLine."No." := WhseActivHeader."No.";
  WhseActivLine."Line No." := numLinea;
  WhseActivLine."Action Type":= WhseActivLine."Action Type"::Take;
  WhseActivLine."Location Code" := CabeceraBulto.Almacen;
  WhseActivLine."Zone Code" := CabeceraBulto."Zona Actual";
  WhseActivLine."Bin Code" := CabeceraBulto."Ubicacion Actual";
  WhseActivLine."Item No." := LineasBulto.Producto;
  WhseActivLine.Description := Producto.Description;
  WhseActivLine."Unit of Measure Code" := LineasBulto."Ud Medida";
  WhseActivLine."Qty. per Unit of Measure" := LineasBulto."Cantidad Por Ud Medida";
  WhseActivLine."Starting Date" := WORKDATE;
  WhseActivLine."Bin Ranking" := Bin."Bin Ranking";
  WhseActivLine."Bin Type Code" := Bin."Bin Type Code";
  WhseActivLine.VALIDATE(Quantity, LineasBulto."Cantidad (Base)");
  WhseActivLine."Qty. (Base)" := LineasBulto."Cantidad (Base)";
  WhseActivLine."Qty. to Handle" := LineasBulto."Cantidad (Base)";
  WhseActivLine."Qty. to Handle (Base)" := LineasBulto."Cantidad (Base)";
  WhseActivLine."Qty. Outstanding" := LineasBulto."Cantidad (Base)";
  WhseActivLine."Qty. Outstanding (Base)" := LineasBulto."Cantidad (Base)";
  WhseActivLine."Source No." := LineasBulto."No Bulto";
  WhseActivLine."Source Line No." := LineasBulto."No Linea";
  WhseActivLine."Whse. Document Type" := WhseActivLine."Whse. Document
Type"::"Movement Worksheet";
  WhseActivLine."Whse. Document No." := LineasBulto."No Bulto";
  WhseActivLine."Whse. Document Line No." := LineasBulto."No Linea";

  IF (sBulto <> '') THEN
  BEGIN
     WhseActivLine."Destino Linea" := WhseActivLine."Destino Linea"::Bulto;
     WhseActivLine."Whse. Document Sub Line No." := LineasBulto."No Linea";
     WhseActivLine."No Bulto" := sBulto;
     IF (CabeceraBulto."Tipo Bulto" = CabeceraBulto."Tipo Bulto"::Reserva) THEN
        WhseActivLine.Trilateral := TRUE;
  END ELSE BEGIN
     WhseActivLine."Destino Linea" := WhseActivLine."Destino Linea"::Ubicación;
  END;
  WhseActivLine."FechaHora Creacion" := CURRENTDATETIME;
  WhseActivLine.INSERT;
  numLinea += 10000;

  WhseActivLine.RESET;
  WhseActivLine.INIT;
  WhseActivLine."Activity Type" := WhseActivLine."Activity Type"::Movement;
  WhseActivLine."No." := WhseActivHeader."No.";
  WhseActivLine."Line No." := numLinea;
```

```
  WhseActivLine."Action Type":= WhseActivLine."Action Type"::Place;
  WhseActivLine."Location Code" := CabeceraBulto.Almacen;
  WhseActivLine."Zone Code" := CabeceraBulto."Zona Destino";
  WhseActivLine."Bin Code" := CabeceraBulto."Ubicacion Destino";
  WhseActivLine."Item No." := LineasBulto.Producto;
  WhseActivLine.Description := Producto.Description;
  WhseActivLine."Unit of Measure Code" := LineasBulto."Ud Medida";
  WhseActivLine."Qty. per Unit of Measure" := LineasBulto."Cantidad Por Ud Medida";
  WhseActivLine."Starting Date" := WORKDATE;
  WhseActivLine."Bin Ranking" := Bin."Bin Ranking";
  WhseActivLine."Bin Type Code" := Bin."Bin Type Code";
  WhseActivLine.VALIDATE(Quantity, LineasBulto."Cantidad (Base)");
  WhseActivLine."Qty. (Base)" := LineasBulto."Cantidad (Base)";
  WhseActivLine."Qty. to Handle" := LineasBulto."Cantidad (Base)";
  WhseActivLine."Qty. to Handle (Base)" := LineasBulto."Cantidad (Base)";
  WhseActivLine."Qty. Outstanding" := LineasBulto."Cantidad (Base)";
  WhseActivLine."Qty. Outstanding (Base)" := LineasBulto."Cantidad (Base)";
  WhseActivLine."Source No." := LineasBulto."No Bulto";
  WhseActivLine."Source Line No." := LineasBulto."No Linea";
  WhseActivLine."Whse. Document Type" := WhseActivLine."Whse. Document
Type"::"Movement Worksheet";
  WhseActivLine."Whse. Document No." := LineasBulto."No Bulto";
  WhseActivLine."Whse. Document Line No." := LineasBulto."No Linea";

  IF (sBulto <> '') THEN
  BEGIN
     WhseActivLine."Destino Linea" := WhseActivLine."Destino Linea"::Bulto;
     WhseActivLine."Whse. Document Sub Line No." := LineasBulto."No Linea";
     WhseActivLine."No Bulto" := sBulto;
     IF (CabeceraBulto."Tipo Bulto" = CabeceraBulto."Tipo Bulto"::Reserva) THEN
        WhseActivLine.Trilateral := TRUE;
  END ELSE BEGIN
     WhseActivLine."Destino Linea" := WhseActivLine."Destino Linea"::Ubicación;
  END;
  WhseActivLine."FechaHora Creacion" := CURRENTDATETIME;
  WhseActivLine.INSERT;
  numLinea += 10000;

  ContenidoUbicacion.RESET;
  ContenidoUbicacion.SETRANGE(ContenidoUbicacion."Location
Code",CabeceraBulto.Almacen);
  ContenidoUbicacion.SETRANGE(ContenidoUbicacion."Bin Code",Ubicaciones.Code);
  ContenidoUbicacion.SETRANGE(ContenidoUbicacion."Item No.",LineasBulto.Producto);
  ContenidoUbicacion.SETRANGE(ContenidoUbicacion."Unit of Measure
Code",LineasBulto."Ud Medida");
  IF NOT ContenidoUbicacion.FINDSET(TRUE) THEN
  BEGIN
    ContenidoUbicacion.VALIDATE("Location Code", CabeceraBulto.Almacen);
    ContenidoUbicacion.VALIDATE("Bin Code", Ubicaciones.Code);
    ContenidoUbicacion.VALIDATE("Zone Code", Ubicaciones."Zone Code");
    ContenidoUbicacion.VALIDATE("Item No.", LineasBulto.Producto);
    ContenidoUbicacion."No. Bulto" := sBulto;
    ContenidoUbicacion.INSERT(TRUE);
    ContenidoUbicacion.VALIDATE("Zone Code", Ubicaciones."Zone Code");
```

```
    ContenidoUbicacion.VALIDATE("Bin Code", Ubicaciones.Code);
    ContenidoUbicacion."No. Bulto" := LineasBulto."No Bulto";
    ContenidoUbicacion.MODIFY(TRUE);
  END;
UNTIL LineasBulto.NEXT = 0;

EXIT (TRUE);
```

ReaprovisionarPicking (sBulto)

PseudoCódigo

```
TablaBultos se inicializa
TablaBultos se filtra NoBulto sea sBulto
Si encuentra registros TablaBultos entonces
Repetir
    TablaProducto se inicializa
    TablaProductos se filtra No sea TablaBultos.Producto
    Si no encuentra registros TablaProductos entonces
        ERROR (‘No existe producto’);
    sProd=TablaProductos.No
   dReaprovisionar=TablaBultos.Cantidad (base)
   TablaContenidoU se inicializa
   TablaContenidoU se filtra Codigo Almacen sea CENTRAL
   TablaContenidoU se filtra CodigoUbicacion sea distinto de ISLA Y RTOMASIVO
   TablaContenidoU se filtra CodigoZona sea PICKING
   TablaContenidoU se filtra NoProducto sea sProd
   Si encuentra registros TablaContenidoU entonces
      sUbicacion=TablaContenidoU.CodigoUbicacion
   Sino
      Si  NO TablaSubFamilia.coge(‘SUBFAMILIA’,TablaProducto.CodigoSubFamilia)
          ERROR(‘No existe SUBFAMILIA’)
      Si TablaSubFamilia.UbicacionPickingGenerica=’’ entonces
          ERROR(‘No tiene configurado pasillo’);
      TablaContenido se inicializa
      TablaContenido se filtra CodigoAlmacen sea CENTRAL
      TablaContenido se filtra Codigo sea TablaSubFamilia.UbicacionPickingGenerica
      Si no encuentra registros TablaContenido entonces
         ERROR(‘No existe ubicación);
      sUbicacion=TablaSubFamilia.UbicacionPickingGenerica
     CrearPickingReaprovisionamient(sbulto,sProd,’PICKING’,sUbicacion,
     dReaprovisionar)

    TablaTemporal se inicializa
    TablaTemporal se filtra Producto sea TablaBultos.Producto
    Si encuentra registros TablaTemporal entonces
       Temporal.CantidadServida+=dReaprovisionar
       Modifica Registro TablaTemporal
    RellenarTablaPropuestaBultos(sbulto)
Hasta registro TablaBultos sea vacío
```

Código Original

```
BultoL.RESET;
BultoL.SETRANGE(BultoL."No Bulto", sBulto);
```

```
IF BultoL.FINDSET THEN
REPEAT
   Producto.RESET;
   Producto.SETRANGE("No.", BultoL.Producto);
   IF (NOT Producto.FINDFIRST) THEN
     ERROR('No encuentra el producto ' + Producto."No.");

   sProd:=Producto."No.";
   dReaprovisionar:=BultoL."Cantidad (Base)";

   BinContent.RESET;
   BinContent.SETCURRENTKEY("Location Code","Bin Code","Item No.","Variant
Code","Unit of Measure Code");
   BinContent.SETRANGE("Location Code", 'CENTRAL');
   BinContent.SETFILTER("Bin Code",'<>ISLA&<>RTOMASIVO');
   BinContent.SETRANGE("Zone Code", 'PICKING');
   BinContent.SETRANGE("Item No.", sProd);
   IF BinContent.FINDFIRST THEN
     sUbicacion:=BinContent."Bin Code"
   ELSE
   BEGIN

     IF NOT rcdSubfamilia.GET('SUBFAMILIA',Producto."Global Dimension 1 Code") THEN
       ERROR('NO EXISTE LA FAMILIA %1 DEL PRODUCTO %2 DEL BULTO
%3',Producto."Global Dimension 1 Code",Producto."No.",sBulto);

     IF rcdSubfamilia."Ubicación Picking Genérica"='' THEN
       ERROR('La familia %1 del producto %2, del bulto %3 no tiene configurado ningún
pasillo',
             Producto."Global Dimension 1 Code",Producto."No.",sBulto);

     Bin.RESET;
     Bin.SETRANGE("Location Code",'CENTRAL');
     Bin.SETRANGE(Code,rcdSubfamilia."Ubicación Picking Genérica");
     IF NOT Bin.FINDFIRST THEN
       ERROR('La Ubicación %1 no existe',rcdSubfamilia."Ubicación Picking Genérica");

     sUbicacion:=rcdSubfamilia."Ubicación Picking Genérica";
   END;

   CrearPickingReaprovisionamient(sBulto, sProd, 'PICKING', sUbicacion, dReaprovisionar);

   Temporal.RESET;
   Temporal.SETRANGE(Producto, BultoL.Producto);
   IF Temporal.FINDFIRST THEN
   BEGIN
     Temporal."Cantidad Servida" += dReaprovisionar;
     Temporal.MODIFY;
   END;
   RellenarTablaPropuestaBultos (sBulto);
UNTIL BultoL.NEXT = 0;
```

CrearPickingReaprovisionamient (sBulto, sProd, 'PICKING', sUbicacion, dReaprovisionar)

PseudoCódigo

```
TablaCabActividad se inicializa
TablaCabActividad se filtra Tipo sea Picking
TablaCabActividad se filtra NoDoc sea sbulto
Si encuentra registros TablaCabActividad entonces
    TablaLinActividad se inicializa
    TablaLinActividad se filtra TipoActividad sea TablaCabActividad.Tipo
    TablaLinActividad se filtra No sea TablaCabActividad.No
    Si encuentra ultimo registro TablaLinActividad entonces
       numLinea:=numLinea+10000
Sino
    ConfigAlmacen se inicializa
    ConfigAlmacen se coge primer registro
    TablaCabActividad se inicializa
    TablaCabActividad.No=''
    TablaCabActividad.Tipo=Picking
    TablaCabActividad.CodigoAlmacen:=CENTRAL
    TablaCabActividad.MetodoOrdenacion=''
    TablaCabActividad.NoDoc=sbulto
    TablaCabActividad.NoSeriesRegistro=ConfigAlmacen.NosRegistroAlmacenPick
    Insertar Registro TablaCabActividad

    TablaBultos se inicializa
    TablaBultos se filtra NoBulto sea sbulto
   Si no encuentra registros TablaBultos entonces
      ERROR ('No existe el bulto');

   TablaLinBultos se inicializa
   TablaLinBultos se filtra NoBulto sea sbulto
   TablaLinBultos se filtra Producto sea sProducto
   Si encuentra registros TablaLinBultos entonces
       TablaLinActividad se inicializa
       TablaLinActividad.TipoActividad=TablaCabActividad.Tipo
       TablaLinActividad.No=TablaCabActividad.No
       TablaLinActividad.TipoAccion=Traer
       TablaLinActividad.CodigoAlmacen=TablaBulto.Almacen
       TablaLinActividad.NoProducto=TablaLinBulto.Producto
       TablaLinActividad.UdMedida=TablaLinBulto.UdMedida
       TablaLinActividad.CantUdMedida=TablaLinBultos.CantUdMedida
       TablaLinActividad.Descripcion='REAPROVISIONAMIENTO BULTO';
       TablaLinActividad.Reaprovisionamiento=Verdadero
       TablaLinActividad.TipoDocAlmacen=HojaMovimiento
       TablaLinActividad.NoDocAlmacen=sbulto
       TablaLinActividad.NoLinDocAlmacen=0
       TablaLinActividad.NoOrigen=sbulto
       TablaLinActividad.DestinoLinea=Bulto
       TablaLinActividad.NoBulto=sbulto
       TablaLinActividad.Prioridad=sPrioridad

       TablaContenido se inicializa
```

TablaContenido se filtra Codigo Almacen sea TablaBultos.Almacen
TablaContenido se filtra Codigo sea ISLA
Si no encuentra registros TablaContenido entonces
  ERROR (‘No encuentra ubicación’)

TablaLinActividad.CodigoZona=TablaContenido.CodigoZona
TablaLinActividad.CodigoUbicacion=TablaContenido.CodigoUbicacion
TablaLinActividad.RankingUbicacion=TablaContenido.RankingUbicacion
TablaLinActividad.TipoCodigoUbicacion=TablaContenido.TipoCodigoUbicacion
TablaLinActividad.Cantidad (base)=dCantidad
TablaLinActividad.CantidadPendiente (base)=dCantidad
TablaLinActividad.CantidadaManipular=0
TablaLinActividad.CantidadaManipular (base)=0
TablaLinActividad.CantidadManipulada=0
TablaLinActividad.FechaHoraCreacion=Hoy
Insertar Registro TablaLinActividad

numLinea=numLinea+10000

TablaLinActividad se inicializa
TablaLinActividad.TipoActividad=TablaCabActividad.Tipo
TablaLinActividad.No=TablaCabActividad.No
TablaLinActividad.TipoAccion=Colocar
TablaLinActividad.CodigoAlmacen=TablaBulto.Almacen
TablaLinActividad.NoProducto=TablaLinBulto.Producto
TablaLinActividad.UdMedida=TablaLinBulto.UdMedida
TablaLinActividad.CantUdMedida=TablaLinBultos.CantUdMedida
TablaLinActividad.Descripcion=’REAPROVISIONAMIENTO BULTO’;
TablaLinActividad.Reaprovisionamiento=Verdadero
TablaLinActividad.TipoDocAlmacen=HojaMovimiento
TablaLinActividad.NoDocAlmacen=sbulto
TablaLinActividad.NoLinDocAlmacen=0
TablaLinActividad.NoOrigen=sbulto
TablaLinActividad.DestinoLinea=Bulto
TablaLinActividad.NoBulto=sbulto
TablaLinActividad.Prioridad=sPrioridad

TablaContenido se inicializa
TablaContenido se filtra Codigo Almacen sea TablaBultos.Almacen
TablaContenido se filtra Codigo sea sUbicacion
Si no encuentra registros TablaContenido entonces
  ERROR (‘No encuentra ubicación’)

TablaLinActividad.CodigoZona=TablaContenido.CodigoZona
TablaLinActividad.CodigoUbicacion=TablaContenido.CodigoUbicacion
TablaLinActividad.RankingUbicacion=TablaContenido.RankingUbicacion
TablaLinActividad.TipoCodigoUbicacion=TablaContenido.TipoCodigoUbicacion
TablaLinActividad.Cantidad (base)=dCantidad
TablaLinActividad.CantidadPendiente (base)=dCantidad
TablaLinActividad.CantidadaManipular=0
TablaLinActividad.CantidadaManipular (base)=0
TablaLinActividad.CantidadManipulada=0
TablaLinActividad.FechaHoraCreacion=Hoy
Insertar Registro TablaLinActividad

Código Original

```
WhseActivHeader.RESET;
WhseActivHeader.SETRANGE(WhseActivHeader.Type, WhseActivHeader.Type::Pick);
WhseActivHeader.SETRANGE(WhseActivHeader."External Document No.", sBulto);
IF WhseActivHeader.FINDFIRST THEN
BEGIN
   WhseActivLine.RESET;
   WhseActivLine.SETRANGE(WhseActivLine."Activity Type",     WhseActivHeader.Type);
   WhseActivLine.SETRANGE(WhseActivLine."No.", WhseActivHeader."No.");
   IF WhseActivLine.FINDLAST THEN
      numLinea := WhseActivLine."Line No." + 10000;
END ELSE BEGIN
   WarehouseSetup.RESET;
   WarehouseSetup.GET;
   WhseActivHeader.RESET;
   WhseActivHeader.INIT;
   WhseActivHeader."No." := '';
   WhseActivHeader.Type := WhseActivHeader.Type::Pick;
   WhseActivHeader."Location Code" := InforEmpresa."Location Code";
   WhseActivHeader."Sorting Method" := WhseActivHeader."Sorting Method"::' ';
   WhseActivHeader."External Document No." := sBulto;
   WhseActivHeader."Registering No. Series" := WarehouseSetup."Registered Whse. Pick
Nos.";
   WhseActivHeader.INSERT(TRUE);

Bulto.RESET;
Bulto.SETRANGE("No Bulto", sBulto);
IF NOT Bulto.FINDFIRST THEN
  ERROR('No encuentra el bulto ' + sBulto);

BultoLinea.RESET;
BultoLinea.SETRANGE("No Bulto", sBulto);
BultoLinea.SETRANGE(Producto, sProducto);
IF BultoLinea.FINDFIRST THEN
BEGIN
   WhseActivLine.RESET;
   WhseActivLine.INIT;
   WhseActivLine."Activity Type" := WhseActivHeader.Type;
   WhseActivLine."No." := WhseActivHeader."No.";
   WhseActivLine."Line No." := numLinea;
   WhseActivLine."Action Type" := WhseActivLine."Action Type"::Take;
   WhseActivLine."Location Code" := Bulto.Almacen;
   WhseActivLine.VALIDATE(WhseActivLine."Item No.", BultoLinea.Producto);
   WhseActivLine.VALIDATE(WhseActivLine."Unit of Measure Code", BultoLinea."Ud
Medida");
   WhseActivLine.VALIDATE(WhseActivLine."Qty. per Unit of Measure",
BultoLinea."Cantidad Por Ud Medida");
   WhseActivLine.Description := 'REAPROVISIONAMIENTO BULTO';
   WhseActivLine.Reaprovisionamiento := TRUE;
   WhseActivLine."Whse. Document Type" := WhseActivLine."Whse. Document
Type"::"Movement Worksheet";
   WhseActivLine."Whse. Document No." := sBulto;
```

```
    WhseActivLine."Whse. Document Line No." := 0;
    WhseActivLine."Source No." := sBulto;
    WhseActivLine."Destino Linea" := WhseActivLine."Destino Linea"::Bulto;
    WhseActivLine."No Bulto" := sBulto;
    WhseActivLine.Prioridad := sPrioridad;

    Bin.RESET;
    Bin.SETRANGE(Bin."Location Code", Bulto.Almacen);
    Bin.SETRANGE(Bin.Code, 'ISLA');
    IF NOT Bin.FINDFIRST THEN
      ERROR('No encuentra la ubicación.');
    WhseActivLine."Zone Code" := Bin."Zone Code";
    WhseActivLine."Bin Code" := Bin.Code;
    WhseActivLine."Bin Ranking" := Bin."Bin Ranking";
    WhseActivLine."Bin Type Code" := Bin."Bin Type Code";
    WhseActivLine.VALIDATE(WhseActivLine."Qty. (Base)", dCantidad);
    WhseActivLine.VALIDATE(WhseActivLine."Qty. Outstanding (Base)", dCantidad);
    WhseActivLine.VALIDATE(WhseActivLine."Qty. to Handle (Base)", 0);
    WhseActivLine.VALIDATE(WhseActivLine."Qty. to Handle", 0);
    WhseActivLine.VALIDATE(WhseActivLine."Qty. Handled (Base)", 0);
    WhseActivLine."FechaHora Creacion" := CURRENTDATETIME;
    WhseActivLine.INSERT(TRUE);
    numLinea := numLinea + 10000;

    WhseActivLine.RESET;
    WhseActivLine.INIT;
    WhseActivLine."Activity Type" := WhseActivHeader.Type;
    WhseActivLine."No." := WhseActivHeader."No.";
    WhseActivLine."Line No." := numLinea;
    WhseActivLine."Action Type" := WhseActivLine."Action Type"::Place;
    WhseActivLine."Location Code" := Bulto.Almacen;
    WhseActivLine.VALIDATE(WhseActivLine."Item No.", BultoLinea.Producto);
    WhseActivLine.VALIDATE(WhseActivLine."Unit of Measure Code", BultoLinea."Ud
Medida");
    WhseActivLine.VALIDATE(WhseActivLine."Qty. per Unit of Measure",
BultoLinea."Cantidad Por Ud Medida");
    WhseActivLine.Description := 'REAPROVISIONAMIENTO BULTO';
    WhseActivLine.Reaprovisionamiento := TRUE;
    WhseActivLine."Source No." := sBulto;
    WhseActivLine."Whse. Document Type" := WhseActivLine."Whse. Document
Type"::"Movement Worksheet";
    WhseActivLine."Whse. Document No." := sBulto;
    WhseActivLine."Whse. Document Line No." := 0;
    WhseActivLine."Destino Linea" := WhseActivLine."Destino Linea"::Bulto;
    WhseActivLine."No Bulto" := sBulto;
    WhseActivLine.Prioridad := sPrioridad;
    Bin.RESET;
    Bin.SETRANGE(Bin."Location Code", Bulto.Almacen);
    Bin.SETRANGE(Bin.Code, sUbicacion);
    IF NOT Bin.FINDFIRST THEN
      ERROR('No encuentra la ubicación ' + sUbicacion +'.');
    WhseActivLine."Zone Code" := Bin."Zone Code";
    WhseActivLine."Bin Code" := Bin.Code;
    WhseActivLine."Bin Ranking" := Bin."Bin Ranking";
```

```
    WhseActivLine."Bin Type Code" := Bin."Bin Type Code";
    WhseActivLine.VALIDATE(WhseActivLine."Qty. (Base)", dCantidad);
    WhseActivLine.VALIDATE(WhseActivLine."Qty. Outstanding (Base)", dCantidad);
    WhseActivLine.VALIDATE(WhseActivLine."Qty. to Handle (Base)", 0);
    WhseActivLine.VALIDATE(WhseActivLine."Qty. to Handle", 0);
    WhseActivLine.VALIDATE(WhseActivLine."Qty. Handled (Base)", 0);
    WhseActivLine."FechaHora Creacion" := CURRENTDATETIME;
    WhseActivLine.INSERT(TRUE);
END;
```

RellenarTablaPropuestaBultos (sBulto)

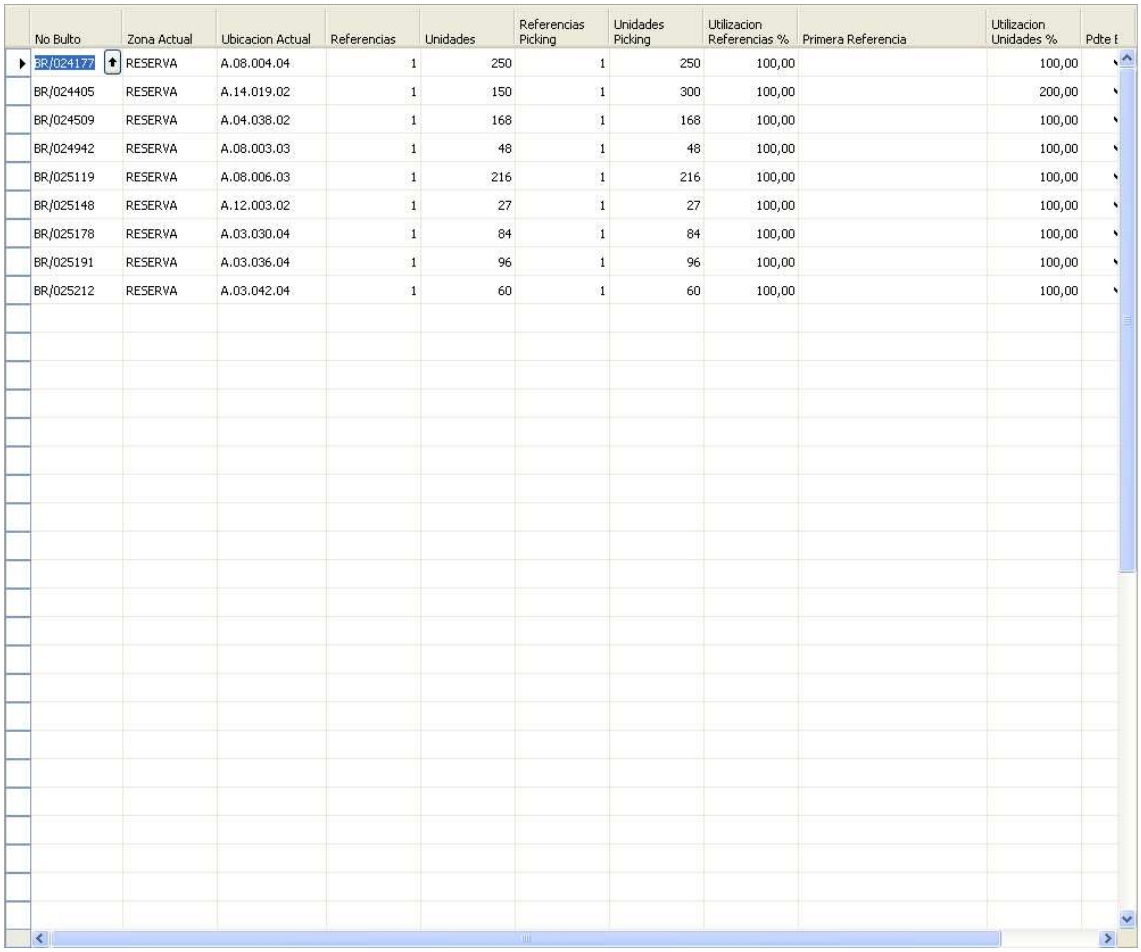
PseudoCódigo

```
TablaBultos se inicializa
TablaBultos se filtra NoBulto sea sBulto
Si no encuentra registros TablaBultos entonces
   ERROR (‘No existe Bulto’);
Se calcula TablaBultos.CantidadProducto
TablaLinBultos se inicializa
TablaLinBultos se filtra NoBulto sea sBulto
TablaPropuesta se inicializa
TablaPropuesta se filtra NoBulto sea sBulto
Si no encuentra registros TablaPropuesta entonces
   TablaPropuesta se inicializa
   TablaPropuesta.NoBulto=sBulto
   Se inserta registro TablaPropuesta
TablaPropuesta.ZonaActual=TablaBultos.ZonaActual
TablaPropuesta.UbicacionActual=TablaBultos.UbicacionActual
TablaPropuesta.Referencias=TablaLinBultos.NumeroRegistros
TablaPropuesta.Unidades=TablaBultos.CantidadProducto
refPicking=0
udsPicking=0
Si encuentra registros TablaLinBultos entonces
Repetir
  Se calcula TablaLinBultos.CantidadPicking
  Si TablaLinBultos.CantidadPicking>0 entonces
      refPicking+=1;
      udsPicking+=TablaLinBulto.CantidadPicking
Hasta registro TablaLinBulto sea vacío
TablaPropuesta.ReferenciasPicking=refPicking;
TablaPropuesta.UnidadesPicking=udsPicking
TablaPropuesta.UtilizacionRef%=TablaPropuesta.ReferenciasPicking*100/
TablaPropuesta.Referencias
TablaPropuesta.UtilizacionUds%=TablaPropuesta.UnidadesPicking*100/
TablaPropuesta.Unidades
TablaPropuesta.PdteBajar=Verdadero
Modifica registro TablaPropuesta
```

Código Original

```
BultosH.RESET;
BultosH.SETRANGE(BultosH."No Bulto", sBulto);
IF NOT BultosH.FINDFIRST THEN
   ERROR('No encuentra el bulto ' + sBulto);
```

```
BultosH.CALCFIELDS("Cantidad Producto");

BultosL.RESET;
BultosL.SETRANGE("No Bulto", sBulto);

PropuestaBultos.RESET;
PropuestaBultos.SETRANGE("No Bulto", sBulto);
IF NOT PropuestaBultos.FINDFIRST THEN
BEGIN

   PropuestaBultos.INIT;
   PropuestaBultos."No Bulto" := sBulto;
   PropuestaBultos.INSERT;
END;

PropuestaBultos."Zona Actual" := BultosH."Zona Actual";
PropuestaBultos."Ubicacion Actual" := BultosH."Ubicacion Actual";
PropuestaBultos.Referencias := BultosL.COUNT;
PropuestaBultos.Unidades := BultosH."Cantidad Producto";

refPicking := 0;
udsPicking := 0;
IF BultosL.FINDSET THEN
REPEAT
   BultosL.CALCFIELDS("Cantidad Picking");
   IF (BultosL."Cantidad Picking" > 0) THEN
   BEGIN
      refPicking += 1;
      udsPicking += BultosL."Cantidad Picking";
   END;
UNTIL BultosL.NEXT = 0;

PropuestaBultos."Referencias Picking" := refPicking;
PropuestaBultos."Unidades Picking" := udsPicking;
PropuestaBultos."Utilizacion Referencias %" := PropuestaBultos."Referencias Picking" * 100 /
PropuestaBultos.Referencias;
PropuestaBultos."Utilizacion Unidades %" := PropuestaBultos."Unidades Picking" * 100 /
PropuestaBultos.Unidades;
PropuestaBultos."Pdte Bajar" := TRUE;
PropuestaBultos.MODIFY;
```



No Bulto	Zona Actual	Ubicacion Actual	Referencias	Unidades	Referencias Picking	Unidades Picking	Utilizacion Referencias %	Primera Referencia	Utilizacion Unidades %	Pdte E
BR/024177	RESERVA	A.08.004.04	1	250	1	250	100,00		100,00	
BR/024405	RESERVA	A.14.019.02	1	150	1	300	100,00		200,00	
BR/024509	RESERVA	A.04.038.02	1	168	1	168	100,00		100,00	
BR/024942	RESERVA	A.08.003.03	1	48	1	48	100,00		100,00	
BR/025119	RESERVA	A.08.006.03	1	216	1	216	100,00		100,00	
BR/025148	RESERVA	A.12.003.02	1	27	1	27	100,00		100,00	
BR/025178	RESERVA	A.03.030.04	1	84	1	84	100,00		100,00	
BR/025191	RESERVA	A.03.036.04	1	96	1	96	100,00		100,00	
BR/025212	RESERVA	A.03.042.04	1	60	1	60	100,00		100,00	

Ilustración 29 Tabla Propuesta Bultos

Recepciones

En este apartado, se explica cuáles son los pasos que se realizan para cuando llega un pedido de compra de un proveedor.

En primer lugar, los operarios de almacén deben descargar el camión y colocar en palets la mercancía que van descargando.

Entonces, una vez descargado todo el contenido del camión se debe subir al personal de compras el albarán correspondiente.

Por otro lado, con el albarán en mano, el personal de compras debe crear una nueva recepción. Una vez creada, existe una opcion que es traer líneas de pedido y en este caso, el personal de compras debe acceder a esa opción y traer el pedido correspondiente, como muestra la ilustración 30. Una vez seleccionado, el sistema insertará en la recepcion cada una de las líneas existentes en el pedido. El personal de

compras debe comprobar que las cantidades en la recepción y en el albarán coinciden para cada uno de los productos. En caso de no coincidir, debe modificar esa cantidad para que el personal de almacén cuente únicamente las cantidades que venían en el camión.

Una vez cotejadas las líneas de la recepción, el personal de compras debe cambiar el estado de la recepción para que el personal de almacén pueda iniciar el recuento de la mercancía, como muestra la ilustración 31

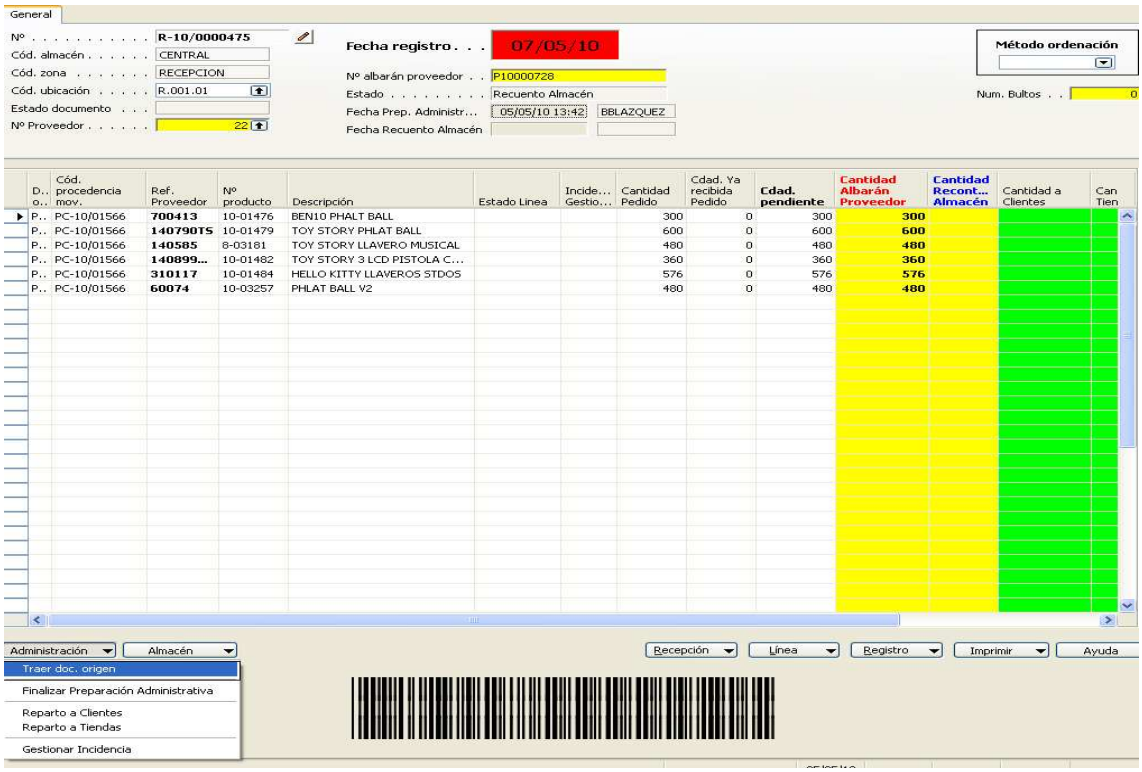


Ilustración 30 Recepción (selección pedido)

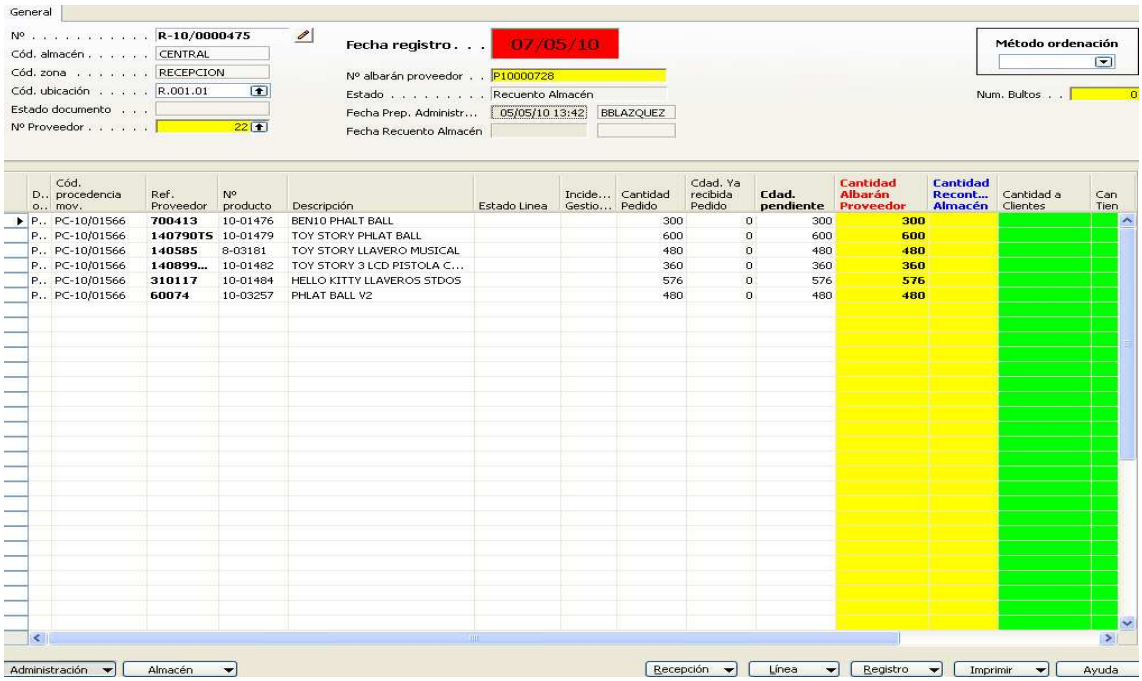


Ilustración 31 Recepción (listo para recontar)

A continuación, el personal de almacén, mediante una PDA, se dispone a recontar la mercancía correspondiente a la recepción. Entonces, el operario de almacén bipa el codigo de barras y la PDA muestra el producto que es y la cantidad teórica que debe tener. El operario en la casilla cantidad marca cuanto ha recontado en ese momento. Este proceso debe hacerlo hasta que no quede nada de ese producto y para cada uno de los productos.

Como se observa en la ilustración 31, en cada línea de la recepción existe un campo que es Estado Línea. Este campo se va actualizando a medida que se hace el conteo de dicho producto. Cuando está en blanco significa que aún no se ha recontado nada, si está En Recuento significa que se ha recontado el producto pero no toda la cantidad del mismo y Recuento OK significa que se ha recontado por completo el producto.

Una vez recontados todos los productos, se debe dar por finalizado el recuento. Para ello, se debe cambiar el estado a la recepción y poner pendiente de registro para que el personal de compras revise si existe alguna incidencia o si el recuento está correcto.

En caso de haber incidencias, se debe saber el motivo de que se haya recontado menos de lo indicado(roturas, caja Standard que viene con unidades de menos, etc..).

Una vez se sepa el motivo de esas incidencias, se deben marcar como incidencias gestionadas para que el sistema permita registrar la recepción

Por último, el personal de compras debe registrar la recepción para que el personal de almacén pueda manipular y colocar los productos de la recepción.

Traer Pedido (TablaCabPedido)

PseudoCódigo

```
TablaLinPedido se inicializa
TablaLinPedido se filtra NumeroDoc sea TablaCabPedido
Si encuentra registros TablaLinPedido
Repetir
    Si TablaLinPedido no se encuentra en ninguna recepcion
        TablaUbicacion se inicializa
        TablaUbicacion coge TablaLinPedido.CodigoAlmacen, Codigo Ubicación
        TablaProducto se inicializa
        Si TablaProducto coge TablaLinPedido.No
            Si TablaUbicacion.TipoClase es disitnto Producto.Clase
                ERROR (‘Ubicaciones distintas’);
        Si CabeceraRecepcion no esta creada
            CrearCabeceraRecepcion
        CrearLíneaRecepcion
Hasta registro TablaLinPedido sea vacio
```

Código Original

```
REPEAT
    IF WhseActivityCreate.CheckIfPurchLine2ReceiptLine("Purchase Line") THEN BEGIN
     // EUCLIDES - EVITAMOS QUE SE TRAIGA PROD CLASE JAULA PARA RECEP NO
JAULA Y VICEVERSA
      Ubicacion.RESET;
      Ubicacion.GET("Location Code", WhseReceiptHeader."Bin Code");
      Producto.RESET;
      IF Producto.GET("Purchase Line"."No.") THEN
      BEGIN
        IF (Ubicacion."Warehouse Class Code" <> Producto."Cod. Clase") THEN
          CurrReport.SKIP;
      END;
      // FIN
      IF NOT OneHeaderCreated AND NOT WhseHeaderCreated THEN
        CreateReceiptHeader;
      WhseActivityCreate.PurchLine2ReceiptLine(WhseReceiptHeader,"Purchase Line");
      LineCreated := TRUE;
    END;
UNTIL rcdLin.next=0;
```

Comprueba Recepción(LineaCompra)

PseudoCódigo

```
LineaCompra se calcula CantidadPendienteRecepcion
Si CantidadPendiente<CantidadPendienteRecepcion entonces
    Devuelve Falso
Devuelve Verdadero
```

Código Original

```
PurchLine.CALCFIELDS("Whse. Outstanding Qty. (Base)");
IF ABS(PurchLine."Outstanding Qty. (Base)") <= ABS(PurchLine."Whse. Outstanding Qty. (Base)") THEN
  EXIT;
EXIT(TRUE);
```

CrearCabeceraRecepcion

PseudoCódigo

Tabla CabRecepcion se inicializa
CabRecepcion.No=’’
CabRecepcion.Almacen=CENTRAL
Bloquea la tabla CabRecepcion
Se inserta el registro Tabla CabRecepcion

Código Original

```
WhseShptHeader.INIT;
WhseShptHeader."No." := '';
WhseShptHeader."Location Code" := "Warehouse Request"."Location Code";
WhseShptLine.LOCKTABLE;
WhseShptHeader.INSERT(TRUE);
```

CrearLíneaRecepcion(CabeceraRecepcion, LineaCompra)

PseudoCódigo

LineaRecepcion se inicializa
LineaRecepcion.No:=CabeceraRecepcion.No
LineaRecepcion.LineaNo:=LineaNo+10000
LineaRecepcion.TipoOrigen:=LineaCompra
LineaRecepcion.SubTipoOrigen=LineaCompra.TipoDocumento
LineaRecepcion.NoOrigen=LineaCompra.NoDocumento
LineaRecepcion.NoLineaOrigen=LineaCompra.NoLinea
LineaRecepcion.CodigoAlmacen=LineaCompra.CodigoAlmacen
LineaRecepcion.NoProducto=LineaCompra.No
LineaRecepcion.CodigoVariante=LineaCompra.CodigoVariante
LineaRecepcion.CodigoUnidadMedida=LineaCompra.CodigoUnidadMedida
LineaRecepcion.CantidadPorUnidadMedida=LineaCompra.CantidadPorUnidadMedida
LineaRecepcion.Descripcion=LineaCompra.Descripcion
LineaRecepcion.Descripcion2=LineaCompra.Descripcion2
LineaRecepcion.CantidadRecibida=LineaCompra.CantidadRecibida
LineaRecepcion.FechaRegistro=LineaCompra.FechaEsperallegada
LineaRecepcion.Cantidad=LineaCompra.Cantidad
LineaRecepcion.FechaInicio=LineaCompra.FechaPlaneadaLlegada
LineaRecepcion.CodigoUbicacion=LineaCompra.CodigoUbicacion
LineaRecepcion.NoProveedor=LineaCompra.NoProveedor
Insertar Registro

Código Original

```
WITH WhseReceiptLine DO BEGIN
  RESET;
  "No." := WhseReceiptHeader."No.";
```

```
  SETRANGE("No.","No.");
//  LOCKTABLE;
  IF FINDLAST THEN;
  INIT;
  "Line No." := "Line No." + 10000;
  "Source Type" := DATABASE::"Purchase Line";
  "Source Subtype" := PurchLine."Document Type";
  "Source No." := PurchLine."Document No.";
  "Source Line No." := PurchLine."Line No.";
  WhseMgt.GetSourceDocument("Source Document","Source Type","Source Subtype");
  "Location Code" := PurchLine."Location Code";
  VALIDATE("Item No.",PurchLine."No.");
  "Variant Code" := PurchLine."Variant Code";
  PurchLine.TESTFIELD("Unit of Measure Code");
  "Unit of Measure Code" := PurchLine."Unit of Measure Code";
  "Qty. per Unit of Measure" := PurchLine."Qty. per Unit of Measure";
  Description := PurchLine.Description;
  "Description 2" := PurchLine."Description 2";
  CASE PurchLine."Document Type" OF
    PurchLine."Document Type"::Order:
      BEGIN
        VALIDATE("Qty. Received",ABS(PurchLine."Quantity Received"));
        "Due Date" := PurchLine."Expected Receipt Date";
      END;
    PurchLine."Document Type"::"Return Order":
      BEGIN
        VALIDATE("Qty. Received",ABS(PurchLine."Return Qty. Shipped"));
        "Due Date" := WORKDATE;
      END;
  END;
  VALIDATE(Quantity,ABS(PurchLine.Quantity));
  "Starting Date" := PurchLine."Planned Receipt Date";
  "Bin Code" := PurchLine."Bin Code";
  // ilr. Añadir el nº proveedor
  "No. Proveedor":=PurchLine."Buy-from Vendor No.";
```

Registrar Recepcion

PseudoCódigo

```
LineasRecepcion se inicializa
LineasRecepcion se filtra No que sea No
LineasRecepcion se filtra CantidadARecibir sea >0
Si encuentra registros LineasRecepcion entonces
Repetir
    Se valida campo UnidadMedida
    TablaSolicitud se inicializa
    TablaSolicitud se filtra tipo sea Entrada
    TablaSolicitud se filtra CodigoAlmacen sea CENTRAL
    TablaSolicitud se filtra TipoOrigen sea LineasRecepcion.TipoOrigen
    TablaSolicitud se filtra SubtipoOrigen sea LineasRecepcion.SubtipoOrigen
    TablaSolicitud se filtra NoOrigen sea LineasRecepcion.NoOrigen
    Si TablaSolicitud.Estado<>Lanzado
        Error ('El pedido no esta lanzado');
Hasta registro LineasRecepcion sea vacio
```

CabRecepcion se inicializa
CabRecepcion se filtra No sea LineasRecepcion.No
Se valida fecha registro
CabRecepcion.CrearCab=Verdadero
Modificar registro CabRecepcion
Repetir
    ValidarLineas
    ValidarCabecera
    RegistrarLineaRecepcion
Hasta registro LineasRecepcion sea vacío
CabRecepRegis se inicializa
CabRecepRegis se filtra NoDocumentoAlmacen sea CabRecepcion.No
Si encuentra registro CabRecepRegis entonces
    CrearDocUbicacion

Código Original

```
WITH WhseRcptLine DO BEGIN
  SETCURRENTKEY("No.");
  SETRANGE("No.","No.");
  SETFILTER("Qty. to Receive",'>0');
  IF FINDSET THEN
    REPEAT
      TESTFIELD("Unit of Measure Code");
      WhseRqst.GET(
        WhseRqst.Type::Inbound,"Location Code","Source Type","Source Subtype","Source
No.");
      IF WhseRqst."Document Status" <> WhseRqst."Document Status"::Released THEN
        ERROR(Text000,"Source Document","Source No.");
    UNTIL NEXT = 0
  ELSE
    ERROR(Text001);

  CounterSourceDocOK := 0;
  CounterSourceDocTotal := 0;
  CounterPutAways := 0;
  CLEAR(CreatePutAway);

  WhseRcptHeader.GET("No.");
  WhseRcptHeader.TESTFIELD("Posting Date");

  WhseRcptHeader."Create Posted Header" := TRUE;
  WhseRcptHeader.MODIFY;
  COMMIT;

  SETCURRENTKEY("No.","Source Type","Source Subtype","Source No.","Source Line
No.");
//  FIND('-');
  FINDSET;
  REPEAT
    SETRANGE("Source Type","Source Type");
    SETRANGE("Source Subtype","Source Subtype");
    SETRANGE("Source No.","Source No.");
    InitSourceDocumentLines(WhseRcptLine);
    InitSourceDocumentHeader;
```

```
    COMMIT;

    CounterSourceDocTotal := CounterSourceDocTotal + 1;
    PostSourceDocument(WhseRcptLine);

    IF FINDLAST THEN;
    SETRANGE("Source Type");
    SETRANGE("Source Subtype");
    SETRANGE("Source No.");
  UNTIL NEXT = 0;
  RRegistrada.RESET;
  RRegistrada.SETCURRENTKEY("Whse. Receipt No.");
  RRegistrada.SETRANGE("Whse. Receipt No.", WhseRcptHeader."No.");
  IF RRegistrada.FINDLAST THEN
     cduUbic.CrearDocUbicacionRepartoRecepc(RRegistrada."No.");

  GetLocation("Location Code");
  PutAwayRequired := Location.RequirePutaway("Location Code");
  IF PutAwayRequired AND NOT Location."Use Put-away Worksheet" THEN
    CreatePutAwayDoc(WhseRcptHeader);

  CLEAR(WMSMgt);
  CLEAR(WhseJnlRegisterLine);
END;
```

ValidarLineas

PseudoCódigo

```
CabCompra se inicializa
CabCompra se filtra TipoDocumento sea LineasRecepcion.SubtipoOrigen
CabCompra se filtra No sea LineasRecepcion.NoOrigen
LinCompra se filtra TipoDocumento sea LineasRecepcion.SubTipoOrigen
LinCompra se filtra NoDocumento sea LineasRecepcion.NoOrigen
Si encuentra registros LinCompra entonces
Repetir
   LinRecepcion se filtra NoLineaOrigen sea LinCompra.NoLinea
   Si encuentra registros LinRecepcion entonces
      Si DocumentoOrigen=PedidoCompra entonces
        ModificarLinea=LinCompra.CantidadaRecibir<>
LinRecepcion.Cantidadarecibir
          Si ModificarLinea entonces
             Validar CantidadaRecibir
     Sino
      ModificarLinea=LinCompra.CantidadaEnviar<>- (LinRecepcion.CantidadaRecibir)
      Si ModificarLinea entonces
          Validar CantidadaEnviar
     Si LinCompra.CodigoUbicacion<> LinRecepcion.CodigoUbicacion entonces
        LinCompra.CodigoUbicacion=LinRecepcion.CodigoUbicacion
        Modificar registro LinCompra
 Sino
      Si LinRecepcion.DocumentoOrigen=PedidoCompra entonces
          ModificarLinea=LinCompra.CantidadaRecibir<>0
          Si ModificarLinea entonces
             Validar CantidadaRecibir
```

```
    Sino
        ModificarLinea=LinCompra.CantidadaEnviar<>0
        Si ModificarLinea entonces
            Validar CantidadaEnviar
  Si ModificarLinea entonces
      Modificar Registro LinCompra
Hasta registro LinCompra sea vacío
```

Código Original

```
WhseRcptLine2.COPY(WhseRcptLine);
WITH WhseRcptLine2 DO BEGIN
      PurchHeader.GET("Source Subtype","Source No.");
      PurchLine.SETRANGE("Document Type","Source Subtype");
      PurchLine.SETRANGE("Document No.","Source No.");
//      IF PurchLine.FIND('-') THEN
      IF PurchLine.FINDSET THEN
        REPEAT
          SETRANGE("Source Line No.",PurchLine."Line No.");
//          IF FIND('-') THEN BEGIN
          IF FINDFIRST THEN BEGIN
            IF "Source Document" = "Source Document"::"Purchase Order" THEN BEGIN
              ModifyLine := PurchLine."Qty. to Receive" <> "Qty. to Receive";
              IF ModifyLine THEN
                PurchLine.VALIDATE("Qty. to Receive","Qty. to Receive")
            END ELSE BEGIN
              ModifyLine := PurchLine."Return Qty. to Ship" <> -"Qty. to Receive";
              IF ModifyLine THEN
                PurchLine.VALIDATE("Return Qty. to Ship",-"Qty. to Receive");
            END;
            IF PurchLine."Bin Code" <> "Bin Code" THEN BEGIN
              PurchLine."Bin Code" := "Bin Code";
              ModifyLine := TRUE;
            END;
          END ELSE
            IF "Source Document" = "Source Document"::"Purchase Order" THEN BEGIN
              ModifyLine := PurchLine."Qty. to Receive" <> 0;
              IF ModifyLine THEN
                PurchLine.VALIDATE("Qty. to Receive",0);
            END ELSE BEGIN
              ModifyLine := PurchLine."Return Qty. to Ship" <> 0;
              IF ModifyLine THEN
                PurchLine.VALIDATE("Return Qty. to Ship",0);
            END;
          IF ModifyLine THEN
            PurchLine.MODIFY;
        UNTIL PurchLine.NEXT = 0;
```

Validar Cabecera

PseudoCódigo

```
  FechaVencimientoAnt=CabCompra.FechaEmision
  Validar CabCompra.FechaRegistro
  Si FechaVencimientoAnt<>CabCompra.FechaEmision entonces
      ComprobarPlazos
  Si FechaVencimientoAnt<>CabCompra.FechaEmision entonces
```

```
      CabCompra.CambioAutomatico=Verdadero
ModificarCabecera=Verdadero
Si CabRecepcion.NoProveedor<>'' entonces
   CabCompra.NoProveedor=CabRecepcion.NoProveedor
   ModificarCabecera=Verdadero
Si ModificarCabecera
    Modificar registro CabCompra
```

Código Original

```
FechaVtoAnt:=PurchHeader."Due Date"; //AÑADIDA
PurchHeader.VALIDATE("Posting Date",WhseRcptHeader."Posting Date");
// -001
IF FechaVtoAnt<>PurchHeader."Due Date" THEN
  PurchHeader.ComprobarPlazos(PurchHeader);
//PurchHeader.ComprobarPlazos(PurchHeader);
// +001
IF FechaVtoAnt<>PurchHeader."Due Date" THEN    //AÑADIDA
  PurchHeader."Cambio Automático de Mes":=TRUE;  //AÑADIDA

PurchRelease.RUN(PurchHeader);
ModifyHeader := TRUE;
END;
IF WhseRcptHeader."Vendor Shipment No." <> '' THEN BEGIN
  PurchHeader."Vendor Shipment No." := WhseRcptHeader."Vendor Shipment No.";
  ModifyHeader := TRUE;
END;
IF ModifyHeader THEN
  PurchHeader.MODIFY;
```

ComprobarPlazos (CabCompra)

PseudoCódigo

```
ImportePendiente=ImporterNoFacturado (CabCompra)
Si ImportePendiente=0 entonces
   Salir;
TerminosPagos se inicializa
TerminosPagos se filtra Codigo sea CabCompra.CodigoTerminosPago
Si no encuentra registros TerminosPagos entonces
   ERROR ('Termino de pago desconocido');
FechaVcto=CabCompra.FechaEmision
TablaPlazos se inicializa
TablaPlazos se filtra CodigoTerminosPago sea TerminosPagos.Codigo
Si no encuentra registros TablaPlazos entonces
   CompruebaPresupuesto (FechaVcto, ImportePendiente)
Sino
   Si TerminosPago.TipoIva<>Proporcional
      Error ('No se contempla opcion IVA no proporcional');
   Repetir
      Testear TablaPlazos. % del Total
      ImportePlazo=ImportePendiente*TablaPlazos. % del Total/100
      ComprobarPresupuesto (FechaVcto, ImportePlazo)
      Calcular FechaVcto
      Ajustar Fecha Emision
   Hasta registro TablaPlazos sea vacío
```

Código Original

```
nvImportePendiente:=rcdPedido.ImporteNoFacturado(rcdPedido);
IF nvImportePendiente = 0 THEN EXIT;

IF NOT TerminosPago.GET(rcdPedido."Payment Terms Code") THEN
  ERROR('Pedido con término de pago desconocido.');

dvFechaVcto:=rcdPedido."Due Date";
rcdPlazos.RESET;
rcdPlazos.SETRANGE("Payment Terms Code",TerminosPago.Code);
IF NOT rcdPlazos.FINDFIRST THEN
BEGIN
 ComprobarPresupuesto(dvFechaVcto,nvImportePendiente);
END
ELSE
BEGIN
 IF TerminosPago."VAT distribution" <> TerminosPago."VAT distribution"::Proportional
THEN
    ERROR('No se encuentra contemplada la distribución de IVA no proporcional.');
 REPEAT
   rcdPlazos.TESTFIELD("% of Total");
   nvImportePlazo :=  nvImportePendiente * rcdPlazos."% of Total" / 100;
   ComprobarPresupuesto(dvFechaVcto,nvImportePlazo);

   dvFechaVcto := CALCDATE(rcdPlazos."Gap between Installments",dvFechaVcto);
   AjusFechVto.PurchAdjustDueDate(dvFechaVcto,rcdPedido."Pay-to Vendor No.");

 UNTIL rcdPlazos.NEXT=0;
END;
```

ImporteNoFacturado (CabCompra)

PseudoCódigo

```
LinCompra se inicializa
LinCompra se filtra TipoDocumento sea CabCompra.TipoDocumento
LinCompra se filtra NoDocumento sea CabCompra.No
LinCompra se filtra Cantidad (Base) sea <>0
Si encuentra registros LinCompra entonces
    Repetir
        Importe=LinCompra.ImportePendiente+ImporteNoFacturado
    Hasta registro LinCompra sea vacío
```

Código Original

```
rcdLinCompra.RESET;
rcdLinCompra.SETRANGE("Document Type",rcdPedido."Document Type");
rcdLinCompra.SETRANGE("Document No.",rcdPedido."No.");
rcdLinCompra.SETFILTER(rcdLinCompra."Quantity (Base)",'<>%1',0);
IF rcdLinCompra.FINDSET() THEN
BEGIN
  REPEAT
    nvTotFacturado+=rcdLinCompra."Outstanding Amount"+rcdLinCompra."Amt. Rcd. Not
Invoiced";
```

```
  UNTIL rcdLinCompra.NEXT=0;
```

ComprobarPresupuesto (nvFecha, ImportePendiente)

PseudoCódigo

```
ConfigConta se posiciona en el primer registro
TablaDimension se inicializa
TablaDimension se filtra CodigoDimension sea ConfigConta.CodigoDimension1
TablaDimension se filtra Codigo sea CabCompra.CodigoDimension1
TablaPresupuestoMes se filtra TipoPresupuesto sea TablaDimension.TipoPresupuesto
TablaPresupuestoMes se filtra FechaInicial sea nvFecha
TablaPresupuestoMes se filtra FechaFinal sea 31/12/9999
Si no encuentra registros TablaPresupuestoMes entonces
   ERROR (‘No existe Presupuesto’);
TablaPresupuestoFam se filtra TipoPresupuesto sea TablaDimension.TipoPresupuesto
TablaPresupuestoFam se filtra NºPptoAño sea TablaPresupuestoMes.NºPptoAño
TablaPresupuestoFam se filtra NºPptoMes sea TablaPresupuestoMes.NºPptoMes
TablaPresupuestoFam se filtra CodFamPresup sea TablaDimension.CodigoDimension1
Si no encuentra registros TablaPresupuestoFam entonces
     Error (‘No existe presupuesto creado’);
Si CabCompra.DePlantilla entonces
      TablaPresupuestoFam se filtra FiltroProveedor sea CabCompra.NoProveedor
      Calcula TablaPresupuestoFam.ImporteResPendiente
      Si TablaPresupuestoFam.ImporteResPendiente<ImportePendiente entonces
            Error(‘Presupuesto SobrePasado’);
Sino
      Calcula TablaPresupuestoFam.ImporteConsumTot
      Si TablaPresupuestoFam.ImporteConsumTot+ImportePendiente>
         TablaPresupuestoFam.ImportePresup entonces
            Error(‘Presupuesto SobrePasado’);
```

Código Original

```
GLSetup.GET;
DimVal.GET(GLSetup."Global Dimension 1 Code","Shortcut Dimension 1 Code");
rcdPresupMes.SETRANGE(rcdPresupMes."Tipo Presupuesto",DimVal."Tipo Presupuesto");
rcdPresupMes.SETRANGE("Fecha Inicial",0D,PARFecha);
rcdPresupMes.SETRANGE("fecha final",PARFecha,31129999D);

IF NOT rcdPresupMes.FINDFIRST THEN
  ERROR('No existe ningún presupuesto creado para %1',FORMAT(PARFecha));

rcdPresupFamilia.SETRANGE(rcdPresupFamilia."Tipo Presupuesto",DimVal."Tipo
Presupuesto");
rcdPresupFamilia.SETRANGE("Nº Ppto Año",rcdPresupMes."Nº Ppto Año");
rcdPresupFamilia.SETRANGE("Nº Ppto Mes",rcdPresupMes."Nº Ppto Mes");
rcdPresupFamilia.SETRANGE("Cód. Familia
Presupuestaria",FunPPTO.DevDimValorPresup("Shortcut Dimension 1 Code"));
IF NOT rcdPresupFamilia.FINDFIRST THEN
  MESSAGE('No existe ningún presupuesto creado para %1 - %2 - %3',
       FORMAT(rcdPresupMes."Nº Ppto Año"),FORMAT(rcdPresupMes."Nº Ppto Mes"),
       FunPPTO.DevDimValorPresup("Shortcut Dimension 1 Code"));
```

```
IF "De Plantilla" THEN
BEGIN
   rcdPresupFamilia.SETFILTER("Filtro Proveedor","Buy-from Vendor No.");
   rcdPresupFamilia.CALCFIELDS("Importe Reservado Pendiente");
   IF rcdPresupFamilia."Importe Reservado Pendiente" < PARImporte THEN
     ERROR('¡¡¡¡¡ ATENCION !!!!!!!\\'+
         'PRESUPUESTO PROGRAMADO PARA '+FORMAT("Buy-from Vendor Name") +'
SOBREPASADO\'+
         '-------------------------------------------------------\'+
         '\Pedido ' + FORMAT("No.") +
         '\Presupuesto ' + FORMAT(rcdPresupFamilia."Nº Ppto Año") +
         '\Mes ' + FORMAT(rcdPresupMes."Descripción Periodo") +
         '\Familia ' + FORMAT(rcdPresupFamilia."Cód. Familia Presupuestaria") +
         '\Saldo Disponible ' + FORMAT(rcdPresupFamilia."Importe Reservado Pendiente") +
         '\Importe Vencimiento ' + FORMAT(PARImporte));
END
ELSE
BEGIN
 rcdPresupFamilia.CALCFIELDS(rcdPresupFamilia."Importe Consumido Total");
 IF rcdPresupFamilia."Importe Consumido Total"+PARImporte > rcdPresupFamilia."Importe
Presupuestado" THEN
   ERROR('¡¡¡¡¡ ATENCION !!!!!!!\\'+
       'PRESUPUESTO SOBREPASADO\'+
       '--------------------------------------\'+
       '\Pedido ' + FORMAT("No.") +
       '\Presupuesto ' + FORMAT(rcdPresupFamilia."Nº Ppto Año") +
       '\Mes ' + FORMAT(rcdPresupMes."Descripción Periodo") +
       '\Familia ' + FORMAT(rcdPresupFamilia."Cód. Familia Presupuestaria") +
       '\Importe Presupuestado ' + FORMAT(rcdPresupFamilia."Importe Presupuestado") +
       '\Importe Consumido ' + FORMAT(rcdPresupFamilia."Importe Consumido Total") +
       '\Importe Vencimiento ' + FORMAT(PARImporte));
END;
```

RegistrarLinRecepcion (LinRecepcion)

PseudoCódigo

```
CabRepecion se filtra No sea LinRecepcion.No
Si LinRecepcion.DocumentoOrigen=PedidoCompra
      CabCompra.Recibir=Verdadero
Sino
      CabCompra.Enviar=Verdadero
      CabCompra.Facturar=Falso
RegistraDocumentoCompra
```

Código Original

```
WITH WhseRcptLine DO BEGIN
 WhseRcptHeader.GET("No.");
     IF "Source Document" = "Source Document"::"Purchase Order" THEN
       PurchHeader.Receive := TRUE
     ELSE
       PurchHeader.Ship := TRUE;
     PurchHeader.Invoice := FALSE;

     PurchPost.SetWhseRcptHeader(WhseRcptHeader);
```

```
    CASE WhseSetup."Receipt Posting Policy" OF
     WhseSetup."Receipt Posting Policy"::"Posting errors are not processed":
      BEGIN
       PurchPost.RUN(PurchHeader);
       CounterSourceDocOK := CounterSourceDocOK + 1;
      END;
     WhseSetup."Receipt Posting Policy"::"Stop and show the first posting error":
      BEGIN
       PurchPost.RUN(PurchHeader);
       CounterSourceDocOK := CounterSourceDocOK + 1;
      END;
    END;
    CLEAR(PurchPost);
   END;
```

CrearDocUbicación (codRecepcion)

PseudoCódigo

```
Insertar=Verdadero;
CabeceraUbicacion se inicializa
CabeceraUbicacion se filtra Tipon sea Ubicar
CabeceraUbicacion se filtra NoDocExterno sea codRecepcion
Si encuentra registro CabeceraUbicacion entonces
      Insertar=Falso
TablaRecepcion se inicializa
TablaRecepcion se filtra No sea codRecepcion
Si no encuentra registros entonces
       Error (‘No existe recepcion’);
TablaReparto se inicializa
TablaReparto se filtra No Recep sea codRecepcion
TablaReparto se filtra Estado sea vacío
Si encuentra registros TablaReparto entonces
    numLinea=10000
    Si insertar entonces
       CabeceraUbicacion se inicializa
       CabeceraUbicacion.Tipo=Ubicación
       CabeceraUbicacion.NoDocExterno=TablaRecepcion.No
       CabeceraUbicacion.CodigoAlmacen=Central
       Insertar Registro
   Sino
       LineaUbicacion se inicializa
      LineaUbicacion se filtra No sea CabeceraUbicacion.No
      Si encuentra ultimo registro entonces
            NumLinea=LineaUbicacion.NoLinea+10000
      Repetir
            LineaRecepcion se inicializa
            LineaRecepcion se filtra No sea TablaReparto.NoRecep
            LineaRecepcion se filtra NoLinea sea TablaReparto.NoLineaRecep
            Si no encuentra registro LineaRecepcion entonces
                 Error (‘No existe línea recepcion’);
            TablaUbicaciones se inicializa
            TablaUbicaciones se filtra Almacen sea LineaRecepcion.Almacen
            TablaUbicaciones se filtra Codigo sea LineaRecepcion.Codigo
            Si no encuentra registro TablaUbicaciones entonces
```

```
    ERROR (‘No existe ubicación’);
 LineaUbicacion se inicializa
 LineaUbicacion.TipoActividad=CabeceraUbicacion.Tipo
 LineaUbicacion.No=CabeceraUbicacion.No
 LineaUbicacion.NoLinea=numLinea
 LineaUbicacion.TipoAccion=Traer
 LineaUbicacion.TipoOrigen=39
LineaUbicacion.SubtipoOrigen=1
LineaUbicacion.NoOrigen=LineaRecepcion.NoOrigen
LineaUbicacion.NoLineaOrigen=LineaRecepcion.NoLineaOrigen
LineaUbicacion.DocumentoOrigen=Pedido Compra
LineaUbicacion.TipoDocAlmacen=Recepcion
LineaUbicacion.NoDocAlmacen=LineaRecepcion.No
LineaUbicacion.NoLineaDocAlmacen=LineaRecepcion.NoLinea
LineaUbicacion.NoSubLineaDocAlmacen=TablaReparto.NoLineaReparto
LineaUbicacion.Almacen=LineaRecepcion.Almacen
LineaUbicacion.NoEstante=LineaRecepcion.NoEstante
LineaUbicacion.FechaVencimiento=LineaRecepcion.FechaVencimiento
LineaUbicacion.FechaInicio=LineaRecepcion.FechaInicio
LineaUbicacion.DestinoLinea=TablaReparto.ProcesoUbicacion
LineaUbicacion.NoProducto=LineaRecepcion.NoProducto
LineaUbicacion.CodUdMedida=TablaReparto.CodUdMedida
LineaUbicacion.CantidadBase=TablaReparto.CantidadBase
LineaUbicacion.CantidadPendBase=TablaReparto.CantidadPendBase
LineaUbicacion.CantidadManipular=0
LineaUbicacion.CantidadManipularBase=0
LineaUbicacion.FechaHoraCreacion=Hoy
LineaUbicacion.CodigoZona=LineaRecepcion.CodigoZona
LineaUbicacion.RankingUbicacion=TablaUbicaciones.RankingUbicacion
LineaUbicacion.TipoUbicacion=TablaUbicaciones.TipoUbicacion
Insertar Registro

NumLinea:=numLinea+10000;
Si TablaReparto.ProcesoUbicacion <> Bulto entonces
    TablaUbicaciones se inicializa
    TablaUbicaciones se filtra Almacen sea TablaReparto.AlmacenDestino
    TablaUbicaciones se filtra Codigo sea TablaReparto.UbicacionDestino
    Si no encuentra registro TablaUbicaciones entonces
        ERROR (‘No se encuentra ubicación’);
 LineaUbicacion se inicializa
 LineaUbicacion.TipoActividad=CabeceraUbicacion.Tipo
 LineaUbicacion.No=CabeceraUbicacion.No
 LineaUbicacion.NoLinea=numLinea
 LineaUbicacion.TipoAccion=Colocar
 LineaUbicacion.TipoOrigen=39
LineaUbicacion.SubtipoOrigen=1
LineaUbicacion.NoOrigen=LineaRecepcion.NoOrigen
LineaUbicacion.NoLineaOrigen=LineaRecepcion.NoLineaOrigen
LineaUbicacion.DocumentoOrigen=Pedido Compra
LineaUbicacion.TipoDocAlmacen=Recepcion
LineaUbicacion.NoDocAlmacen=LineaRecepcion.No
LineaUbicacion.NoLineaDocAlmacen=LineaRecepcion.NoLinea
LineaUbicacion.NoSubLineaDocAlmacen=TablaReparto.NoLineaReparto
LineaUbicacion.Almacen=LineaRecepcion.Almacen
```

```
        LineaUbicacion.NoEstante=LineaRecepcion.NoEstante
        LineaUbicacion.FechaVencimiento=LineaRecepcion.FechaVencimiento
        LineaUbicacion.FechaInicio=LineaRecepcion.FechaInicio
        LineaUbicacion.DestinoLinea=TablaReparto.ProcesoUbicacion
        LineaUbicacion.NoProducto=LineaRecepcion.NoProducto
        LineaUbicacion.CodUdMedida=TablaReparto.CodUdMedida
        LineaUbicacion.CantidadBase=TablaReparto.CantidadBase
        LineaUbicacion.CantidadPendBase=TablaReparto.CantidadPendBase
        LineaUbicacion.CantidadManipular=0
        LineaUbicacion.CantidadManipularBase=0
        LineaUbicacion.FechaHoraCreacion=Hoy
        LineaUbicacion.CodigoZona=TablaReparto.ZonaDestino
        Si TablaReparto.ProcesoUbicacion<>Bulto entonces
            LineaUbicacion.CodigoUbicacion=TablaReparto.UbicacionDestino
        LineaUbicacion.RankingUbicacion=TablaUbicaciones.RankingUbicacion
        LineaUbicacion.TipoUbicacion=TablaUbicaciones.TipoUbicacion
        Insertar Registro
        NumLinea:=NumLinea+10000
       TablaReparto.Estado=DocumentoUbicacion
       TablaReparto se modifica
    Hasta siguiente registro TablaReparto sea vacío
```

Código Original

```
insertarCabecera := TRUE;
DocUbicacion.RESET;
DocUbicacion.SETRANGE(DocUbicacion.Type, DocUbicacion.Type::"Put-away");
DocUbicacion.SETRANGE(DocUbicacion."External Document No.", codRecepcion);
IF DocUbicacion.FINDFIRST THEN
  insertarCabecera := FALSE;

Recepcion.RESET;
Recepcion.SETRANGE("No.",codRecepcion);
IF NOT Recepcion.FINDFIRST THEN
  ERROR('Error. No se encuentra la recepción %1',codRecepcion);

Reparto.RESET;
Reparto.SETRANGE(Reparto."No Recep", codRecepcion);
Reparto.SETRANGE(Reparto.Estado, Reparto.Estado::" ");
IF Reparto.FINDSET THEN
BEGIN
   numLinea := 10000;
   IF (insertarCabecera) THEN
   BEGIN
      DocUbicacion.RESET;
      DocUbicacion.INIT;
      DocUbicacion.Type := DocUbicacion.Type::"Put-away";
      DocUbicacion."External Document No." := Recepcion."No.";
      DocUbicacion.VALIDATE(DocUbicacion."Location Code", InforEmpresa."Location
Code");
      DocUbicacion.INSERT(TRUE);
   END ELSE BEGIN
      DocUbicacionLin.RESET;
      DocUbicacionLin.SETRANGE(DocUbicacionLin."No.", DocUbicacion."No.");
      IF DocUbicacionLin.FINDLAST() THEN
```

```
        numLinea := DocUbicacionLin."Line No." + 10000;
    END;
    REPEAT
      RecepcionLinea.RESET;
      RecepcionLinea.SETRANGE("No.",Reparto."No Recep");
      RecepcionLinea.SETRANGE("Line No.",Reparto."No Linea Recep");
      IF NOT RecepcionLinea.FINDFIRST THEN
        ERROR('Error. No encuentro línea recepción %1 -%1',FORMAT(Reparto."No Recep"),FORMAT(Reparto."No Linea Recep"));


      // TRAER
      Bin.RESET;
      Bin.SETRANGE("Location Code",RecepcionLinea."Location Code");
      Bin.SETRANGE(Code,RecepcionLinea."Bin Code");
      IF NOT Bin.FINDFIRST THEN
        ERROR('Error. No se encuentra ubicación %1',RecepcionLinea."Bin Code");

      DocUbicacionLin.RESET;
      DocUbicacionLin.INIT;
      DocUbicacionLin."Activity Type" := DocUbicacion.Type;
      DocUbicacionLin."No." := DocUbicacion."No.";
      DocUbicacionLin."Line No." := numLinea;
      DocUbicacionLin."Action Type" := DocUbicacionLin."Action Type"::Take;
      DocUbicacionLin."Source Type" := 39;
      DocUbicacionLin."Source Subtype" := 1;
      DocUbicacionLin."Source No." := RecepcionLinea."Source No.";
      DocUbicacionLin."Source Line No." := RecepcionLinea."Source Line No.";
      DocUbicacionLin."Source Document" := DocUbicacionLin."Source Document"::"Purchase Order";
      DocUbicacionLin."Whse. Document Type" := DocUbicacionLin."Whse. Document Type"::Receipt;
      DocUbicacionLin."Whse. Document No." := RecepcionLinea."No.";
      DocUbicacionLin."Whse. Document Line No." := RecepcionLinea."Line No.";
      DocUbicacionLin."Whse. Document Sub Line No." := Reparto."No Linea Reparto";
      DocUbicacionLin."Location Code" := RecepcionLinea."Location Code";
      DocUbicacionLin."Shelf No." := RecepcionLinea."Shelf No.";
      DocUbicacionLin."Due Date" := RecepcionLinea."Due Date";
      DocUbicacionLin."Starting Date" := RecepcionLinea."Starting Date";
      //DocUbicacionLin."Breakbulk No." := 0;
      //DocUbicacionLin."Original Breakbulk" := false;
      DocUbicacionLin."Destino Linea" := Reparto."Proceso Ubicación";
      // Producto y Cantidades
      DocUbicacionLin.VALIDATE(DocUbicacionLin."Item No.", RecepcionLinea."Item No.");
      DocUbicacionLin.VALIDATE(DocUbicacionLin."Unit of Measure Code", Reparto."Cod Ud Medida");
      DocUbicacionLin.VALIDATE(DocUbicacionLin."Qty. (Base)", Reparto."Cantidad (base)");
      DocUbicacionLin.VALIDATE(DocUbicacionLin."Qty. Outstanding (Base)", Reparto."Cantidad (base)");
      //DocUbicacionLin.VALIDATE(DocUbicacionLin."Qty. to Handle (Base)", 0);
      //DocUbicacionLin.VALIDATE(DocUbicacionLin."Qty. Handled", 0);
      DocUbicacionLin."Qty. to Handle (Base)" := 0;
```

```
DocUbicacionLin."Qty. to Handle" := 0;
DocUbicacionLin."FechaHora Creacion" := CURRENTDATETIME;

// Ubicaciones
DocUbicacionLin."Zone Code" := RecepcionLinea."Zone Code";
 DocUbicacionLin.VALIDATE(DocUbicacionLin."Bin Code", RecepcionLinea."Bin
Code");
   DocUbicacionLin."Bin Ranking" := Bin."Bin Ranking";
   DocUbicacionLin."Bin Type Code" := Bin."Bin Type Code";
DocUbicacionLin.INSERT(TRUE);
numLinea += 10000;

//COLOCAR
IF (Reparto."Proceso Ubicación" <> Reparto."Proceso Ubicación"::Bulto) THEN
BEGIN
  Bin.RESET;
  Bin.SETRANGE("Location Code",Reparto."Almacen Destino");
  Bin.SETRANGE(Code,Reparto."Ubicacion Destino");
  IF NOT Bin.FINDFIRST THEN
     ERROR('Error. No se encuentra ubicación %1',Reparto."Ubicacion Destino");
END;

DocUbicacionLin.RESET;
DocUbicacionLin.INIT;
DocUbicacionLin."Activity Type" := DocUbicacion.Type;
DocUbicacionLin."No." := DocUbicacion."No.";
DocUbicacionLin."Line No." := numLinea;
DocUbicacionLin."Action Type" := DocUbicacionLin."Action Type"::Place;
DocUbicacionLin."Source Type" := 39;
DocUbicacionLin."Source Subtype" := 1;
DocUbicacionLin."Source No." := RecepcionLinea."Source No.";
DocUbicacionLin."Source Line No." := RecepcionLinea."Source Line No.";
DocUbicacionLin."Source Document" := DocUbicacionLin."Source
Document"::"Purchase Order";
DocUbicacionLin."Whse. Document Type" := DocUbicacionLin."Whse. Document
Type"::Receipt;
DocUbicacionLin."Whse. Document No." := RecepcionLinea."No.";
DocUbicacionLin."Whse. Document Line No." := RecepcionLinea."Line No.";
DocUbicacionLin."Whse. Document Sub Line No." := Reparto."No Linea Reparto";
DocUbicacionLin."Location Code" := Reparto."Almacen Destino";
DocUbicacionLin."Shelf No." := RecepcionLinea."Shelf No.";
DocUbicacionLin."Due Date" := RecepcionLinea."Due Date";
DocUbicacionLin."Starting Date" := RecepcionLinea."Starting Date";
//DocUbicacionLin."Breakbulk No." := 0;
//DocUbicacionLin."Original Breakbulk" := false;
DocUbicacionLin."Destino Linea" := Reparto."Proceso Ubicación";

// Producto y Cantidades
DocUbicacionLin.VALIDATE(DocUbicacionLin."Item No.", RecepcionLinea."Item
No.");
DocUbicacionLin.VALIDATE(DocUbicacionLin."Unit of Measure Code", Reparto."Cod
Ud Medida");
DocUbicacionLin.VALIDATE(DocUbicacionLin."Qty. (Base)", Reparto."Cantidad
(base)");
```

```
      DocUbicacionLin.VALIDATE(DocUbicacionLin."Qty. Outstanding (Base)",
Reparto."Cantidad (base)");
      //DocUbicacionLin.VALIDATE(DocUbicacionLin."Qty. to Handle (Base)", 0);
      //DocUbicacionLin.VALIDATE(DocUbicacionLin."Qty. Handled", 0);
      DocUbicacionLin."Qty. to Handle (Base)" := 0;
      DocUbicacionLin."Qty. to Handle" := 0;
      DocUbicacionLin."FechaHora Creacion" := CURRENTDATETIME;

      // Ubicaciones
      DocUbicacionLin."Zone Code" := Reparto."Zona Destino";
      IF (Reparto."Proceso Ubicación" <> Reparto."Proceso Ubicación"::Bulto) THEN
        DocUbicacionLin.VALIDATE(DocUbicacionLin."BinCode", Reparto."Ubicacion
Destino");
      DocUbicacionLin."Bin Ranking" := Bin."Bin Ranking";
      DocUbicacionLin."Bin Type Code" := Bin."Bin Type Code";
      DocUbicacionLin.INSERT(TRUE);
      numLinea += 10000;
      // Marcamos las lineas de reparto
      Reparto.Estado := Reparto.Estado::"Documento Ubicación";
      Reparto.MODIFY;
   UNTIL Reparto.NEXT = 0;
END;
MESSAGE('Documento creado con éxito: %1', FORMAT(DocUbicacion."No."));
EXIT(DocUbicacion."No.");
```

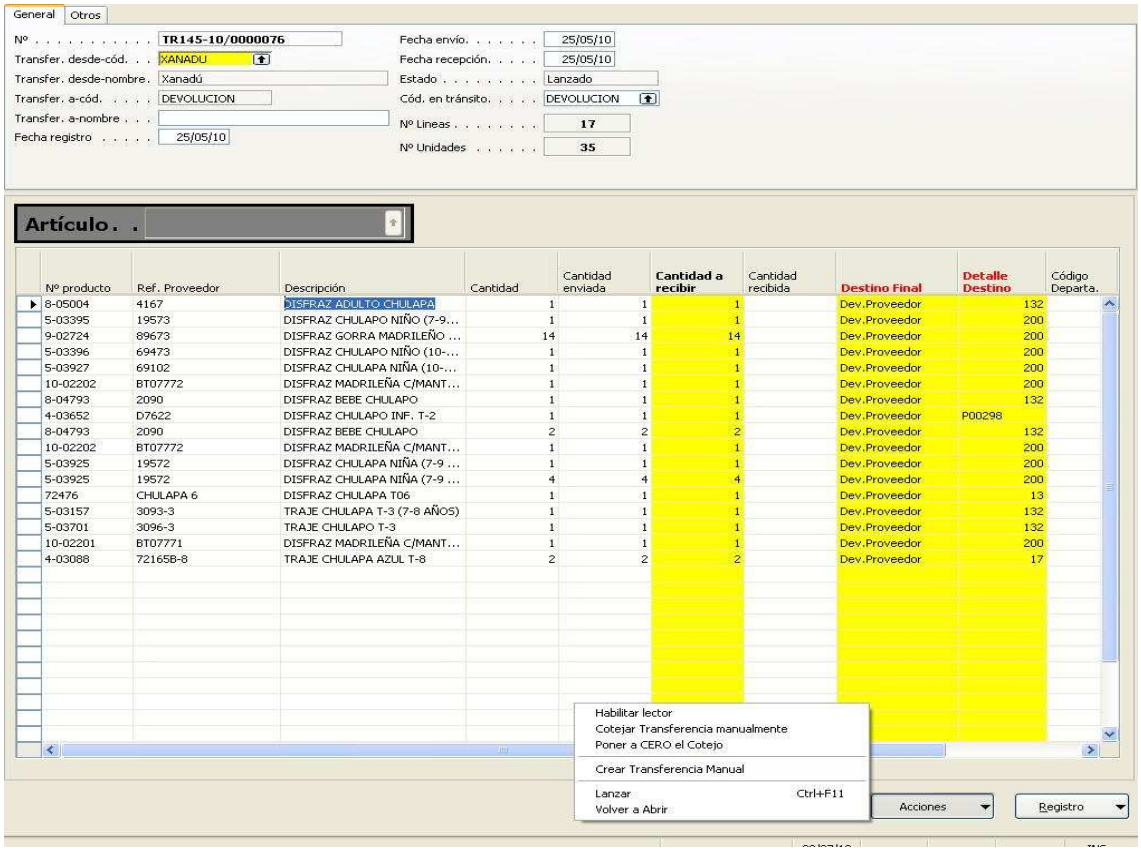
Devoluciones

En este apartado, se explica la forma de trabajo que tienen los operarios de devoluciones. En primer lugar, deben buscar la transferencia correspondiente al palet que tienen delante. Una vez encontrada, como se muestra en la ilustración 32, se debe pulsar la opción de habilitar lector. Esta función lo que provoca es que donde pone ARTÍCULO, se pone activo la barra de al lado y los operarios, mediante el lector de código de barras., puedan pasar los productos. Entonces, el sistema comprueba que el producto leído se encuentra en la transferencia. En caso de que se encuentre, en la columna cantidad a recibir se le añade una unidad. En caso de no encontrarse, el sistema muestra un mensaje de error indicando que dicho producto no se encuentra en la transferencia y para poder insertarlo en la misma debe abrir el pedido e insertar el producto.

Una vez acabado el conteo, los operarios deben registrar la transferencia para que cada producto se registre según el destino que lleve indicado. Según el destino, puede ocurrir lo siguiente:

- Rotura: Se genera un diario que cuando se registra, el producto queda ubicado en el almacén ROTURAS.

- Cross-Docking: Esto ocurre cuando se intercambian productos entre tiendas. Se genera un diario que cuando se registra, el producto queda ubicado en el almacén de la tienda destino
- Picking: Esto ocurre cuando la tienda manda a CENTRAL un producto que tiene por exceso. Al registrar, se genera un documento de ubicación para que los operarios de almacén coloquen esa mercancía en Picking.
- Dev.Proveedor. Esto ocurre cuando la tienda manda un producto que se va a devolver a proveedor. En este caso, se crea un diario y al registrar el producto queda ubicado en el almacén DEVOLPROV. A continuación, ese producto se inserta en un pedido devolución del proveedor correspondiente.
- Taller: Esto ocurre cuando llega un producto que debe reparar taller. Entonces, se genera un diario y al registrar el producto queda ubicado en el almacén TALLER.



Nº producto	Ref. Proveedor	Descripción	Cantidad	Cantidad enviada	Cantidad a recibir	Cantidad recibida	Destino Final	Detalle Destino	Código Departa.
8-05004	4167	DISFRAZ ADULTO CHULAPA	1	1	1		Dev.Proveedor	132	
5-03395	19573	DISFRAZ CHULAPO NIÑO (7-9...	1	1	1		Dev.Proveedor	200	
9-02724	89673	DISFRAZ GORRA MADRILEÑO ...	14	14	14		Dev.Proveedor	200	
5-03396	69473	DISFRAZ CHULAPO NIÑO (10-...	1	1	1		Dev.Proveedor	200	
5-03927	69102	DISFRAZ CHULAPA NIÑA (10-...	1	1	1		Dev.Proveedor	200	
10-02202	BT07772	DISFRAZ MADRILEÑA C/MANT...	1	1	1		Dev.Proveedor	200	
8-04793	2090	DISFRAZ BEBE CHULAPO	1	1	1		Dev.Proveedor	132	
4-03652	D7622	DISFRAZ CHULAPO INF. T-2	1	1	1		Dev.Proveedor	P00298	
8-04793	2090	DISFRAZ BEBE CHULAPO	2	2	2		Dev.Proveedor	132	
10-02202	BT07772	DISFRAZ MADRILEÑA C/MANT...	1	1	1		Dev.Proveedor	200	
5-03925	19572	DISFRAZ CHULAPA NIÑA (7-9 ...	1	1	1		Dev.Proveedor	200	
5-03925	19572	DISFRAZ CHULAPA NIÑA (7-9 ...	4	4	4		Dev.Proveedor	200	
72476	CHULAPA 6	DISFRAZ CHULAPA T06	1	1	1		Dev.Proveedor	13	
5-03157	3093-3	TRAJE CHULAPA T-3 (7-8 AÑOS)	1	1	1		Dev.Proveedor	132	
5-03701	3096-3	TRAJE CHULAPO T-3	1	1	1		Dev.Proveedor	132	
10-02201	BT07771	DISFRAZ MADRILEÑA C/MANT...	1	1	1		Dev.Proveedor	200	
4-03088	72165B-8	TRAJE CHULAPA AZUL T-8	2	2	2		Dev.Proveedor	17	

Ilustración 32 Transferencia de Tienda

RegistrarTransferencia (LineasTransfer)

PseudoCódigo

```
Repetir
    Si LineasTransfer.TransferirA <> DEVOLUCION entonces
        ERROR ();
    TablaProducto se inicializa
    TablaProducto se filtra No sea LineasTransfer.NoProducto
    Si no encuentra registro TablaProducto
        ERROR ('No existe el producto');
    GrabarRecepcion=FALSE
    GrabarPedido=FALSE
    Caso LineasTransfer.DestinoFinal sea
        Rotura: NuevoAlmacen=ROTURAS
        Cross-Docking: NuevoAlmacen=LineasTransfer.DetalleDestino
        Picking: NuevoAlmacen=CENTRAL
                 GrabarRecepcion=TRUE
        Dev.Proveedor: NuevoAlmacen=DEVOLPROV
                       GrabarPedido=TRUE
        Taller: NuevoAlmacen=TALLER

        SeccionDiario se inicializa
        SeccionDiario se filtra NombreSeccion sea TRANSFEREN
        SeccionDiario se filtra Nombre sea USERID
        Si no encuentra Registro SeccionDiario entonces
            SeccionDiario se inicializa
            SeccionDiario.NombreSeccion=TRANSFEREN
            SeccionDiario.Nombre=USERID;
            SeccionDiario.Descripcion='REPARTO TRANSFER TIENDA'
            Insertar Registro SeccionDiario
        LinDiarioProducto se inicializa
        LinDiarioProducto se filtra NombreSeccion sea TRANSFEREN
        LinDiarioProducto se filtra Nombre sea USERID;
        Si encuentra registro LinDiarioProducto entonces
            BorrarTodosLosRegistros
        LinDiarioProducto se inicializa
        LinDiarioProducto.NombreSeccion=TRANSFEREN
        LinDiarioProducto.NombreDiario=USERID
        LinDiarioProducto.TipoMov=Transfer
        LinDiarioProducto.NoLinea=10000
        LinDiarioProducto.NoProducto=LineasTransfer.NoProducto
        LinDiarioProducto.Cantidad=LineasTransfer.CantidadRecibir
        LinDiarioProducto.Almacen=DEVOLUCION
        LinDiarioProducto.FechaRegistro=Hoy
        LinDiarioProducto.NoDocumento=LineasTransfer.NoDocumento
        LinDiarioProducto.NoExterno=LineasTransfer.NoDocumento
        LinDiarioProducto.NoOrdenTransfer=LineasTransfer.NoDocumento
        LinDiarioProducto.Empleado=USERID
        LinDiarioProducto.NuevoAlmacen=NuevoAlmacen
        Insertar registro LinDiarioProducto
        Registrar DiarioProducto
        Si GrabarRecepcion entonces
            SeccionAlmacen se inicializa
            SeccionAlmacen se filtra NombreSeccion sea INVENTARIO
```

```
        SeccionAlmacen se filtra Nombre sea USERID
        Si no encuentra registro SeccionAlmacen entonces
          SeccionAlmacen se inicializa
          SeccionAlmacen.NombreSeccion=INVENTARIO
          SeccionAlmacen.Nombre=USERID
          SeccionAlmacen.Descripcion=Inventario
          SeccionAlmacen.Almacen=CENTRAL
          Insertar registro SeccionAlmacen
      DiarioAlmacen se inicializa
      DiarioAlmacen se filtra NombreSeccion sea INVENTARIO
      DiarioAlmacen se filtra NombreDiario sea USERID
      Si encuentra registros DiarioAlmacen entonces
          BorrarTodosLosRegistros
      DiarioAlmacen se inicializa
      DiarioAlmacen.NombreSeccion=INVENTARIO
      DiarioAlmacen.NombreDiario=USERID
      DiarioAlmacen.Almacen=CENTRAL
      DiarioAlmacen.NoLinea=10000
      DiarioAlmacen.FechaRegistro=HOY
      DiarioAlmacen.NoDocAlmacen=LineasTransfer.NoDocumento
      DiarioAlmacen.NoProducto=LineasTransfer.NoProducto
      DiarioAlmacen.CodigoZona=RECEPCION
      Si TablaProducto.CodClase='' entonces
          DiarioAlmacen.CodigoUbicacion=R.DEVOLU
      Sino
          DiarioAlmacen.CodigoUbicacion=R.DEVOLJ
      DiarioAlmacen.Descripcion=RECEPCION TRANSFERENCIA
      DiarioAlmacen.TipoDocAlmacen=Inventario
      DiarioAlmacen.DesdeCodigoZona=AJUSTE
      DiarioAlmacen.DesdeCodigoUbicacion=AJUSTE
      DiarioAlmacen.Cantidad=LineasTransfer.CantidadRecibir
      RegistrarDiario (DiarioAlmacen)
      CrearDocUbicacion (LineasTransfer)
    Si GrabarPedido entonces
      MeterenPedido (LineasTransfer);
Hasta registro LineasTransfer sea vacío
```

Código Original

```
IF PARLin."Transfer-to Code" <> 'DEVOLUCION' THEN
  EXIT;
rcdItem.RESET;
rcdItem.SETRANGE(rcdItem."No.",PARLin."Item No.");
IF NOT rcdItem.FINDFIRST THEN
  ERROR('No se encuentra el producto %1',FORMAT(PARLin."Item No."));

bGrabarEnRecepcion:=FALSE;
bPedidoProveedor:=FALSE;

CASE PARLin."Destino Final" OF
    PARLin."Destino Final"::Rotura:          cvNuevoAlmacen:='ROTURAS';

    PARLin."Destino Final"::Robo:            cvNuevoAlmacen:='ROBOS';

    PARLin."Destino Final"::"Cross-Docking":  cvNuevoAlmacen:=PARLin."Detalle Destino";
```

```
    PARLin."Destino Final"::Reserva:
      BEGIN
        cvNuevoAlmacen:='CENTRAL';
        bGrabarEnRecepcion:=TRUE;
      END;

    PARLin."Destino Final"::Picking:
      BEGIN
        cvNuevoAlmacen:='CENTRAL';
        bGrabarEnRecepcion:=TRUE;
      END;

    PARLin."Destino Final"::"Dev.Proveedor":
      BEGIN
        cvNuevoAlmacen:='DEVOLPROV';
        bPedidoProveedor:=TRUE;
      END;
    PARLin."Destino Final"::Taller:          cvNuevoAlmacen:='TALLER';

END;
Seccion.RESET;
Seccion.SETRANGE(Seccion."Journal Template Name", 'TRANSFEREN');
Seccion.SETRANGE(Seccion.Name, FORMAT(USERID));
IF NOT Seccion.FINDFIRST THEN
BEGIN
   Seccion.RESET;
   Seccion.INIT;
   Seccion."Journal Template Name" := 'TRANSFEREN';
   Seccion.Name := FORMAT(USERID);
   Seccion.Description := 'REPARTO TRANSFER TIENDA ' + FORMAT(USERID);
   Seccion.INSERT(TRUE)
END;

DiarioProducto.RESET;
DiarioProducto.SETRANGE("Journal Template Name",'TRANSFEREN');
DiarioProducto.SETRANGE("Journal Batch Name",FORMAT(USERID));
IF DiarioProducto.FINDFIRST THEN
  DiarioProducto.DELETEALL;


DiarioProducto.RESET;
DiarioProducto.INIT;
DiarioProducto."Journal Template Name" := 'TRANSFEREN';
DiarioProducto."Journal Batch Name" := FORMAT(USERID);
DiarioProducto."Entry Type" := DiarioProducto."Entry Type"::Transfer;
DiarioProducto."Line No." := 10000;
DiarioProducto.VALIDATE("Item No.", PARLin."Item No.");
DiarioProducto.VALIDATE(Quantity, PARLin."Qty. to Receive");
DiarioProducto.VALIDATE("Location Code", 'DEVOLUCION');
DiarioProducto.VALIDATE("Item No.", PARLin."Item No.");
DiarioProducto.VALIDATE("Posting Date", WORKDATE);

DiarioProducto."Document No." := FORMAT(PARLin."Document No.");
```

```
DiarioProducto."External Document No." := FORMAT(PARLin."Document No.");
DiarioProducto."Transfer Order No.":=FORMAT(PARLin."Document No.");
DiarioProducto."Empleado asignado" := USERID;
DiarioProducto.VALIDATE("New Location Code",cvNuevoAlmacen);

DiarioProducto.INSERT(TRUE);

ItemJnlPostBatch.RUN(DiarioProducto);

IF bGrabarEnRecepcion THEN
BEGIN
   SeccionW.RESET;
   SeccionW.SETRANGE("Journal Template Name",'INVENTARIO');
   SeccionW.SETRANGE(Name,FORMAT(USERID));
   IF NOT SeccionW.FINDFIRST() THEN
   BEGIN
     SeccionW."Journal Template Name" := 'INVENTARIO';
     SeccionW.Name := USERID;
     SeccionW.Description := 'Inventario';
     SeccionW."Location Code":='CENTRAL';
     IF NOT SeccionW.INSERT THEN
        ERROR('Error. Error al insertar sección de diario %1',USERID);
   END;

   LDiario.RESET;
   LDiario.SETRANGE("Journal Template Name",'INVENTARIO');
   LDiario.SETRANGE("Journal Batch Name",FORMAT(USERID));
   IF LDiario.FINDFIRST THEN
     LDiario.DELETEALL;

   LDiario.INIT;
   LDiario.VALIDATE("Journal Template Name",'INVENTARIO');
   LDiario.VALIDATE("Journal Batch Name",FORMAT(USERID));
   LDiario.VALIDATE("Location Code",'CENTRAL');
   LDiario.VALIDATE("Line No.",10000);
   LDiario.VALIDATE("Registering Date",TODAY);
   LDiario.VALIDATE("Whse. Document No.",PARLin."Document No.");

   LDiario.VALIDATE("Item No.",PARLin."Item No.");
   LDiario.VALIDATE("Zone Code",'RECEPCION');
   IF rcdItem."Cod. Clase" = '' THEN
     LDiario.VALIDATE("Bin Code",'R.DEVOLU')
   ELSE
     LDiario.VALIDATE("Bin Code",'R.DEVOLJ');

   LDiario.Description:='RECEPCION TRANSFERENCIA : '+PARLin."Document No.";
   LDiario."Whse. Document Type":=LDiario."Whse. Document Type"::"Whse. Phys.
Inventory";
   LDiario."From Zone Code" := 'AJUSTE';
   LDiario."From Bin Code" := 'AJUSTE';
   LDiario.VALIDATE(Quantity,PARLin."Qty. to Receive");
   CLEAR(CURegistroDiario);
   CURegistroDiario.RUN(LDiario);
  CrearDocUbicacion(PARLin);
```

```
END;
IF bPedidoProveedor THEN
BEGIN
  MeterenPedidoDevolucion(PARLin);
END;
```

MeterenPedido (LineasTransfer)

PseudoCódigo

```
CabeceraCompra se inicializa
CabeceraCompra se filtra TipoDoc sea Pedido
CabeceraCompra se filtra Proveedor sea LineasTransfer.DetalleDestino
CabeceraCompra se filtra Estado sea Abierto
CabeceraCompra se filtra No sea DEV-*
Si no encuentra registro CabeceraCompra entonces
    CabeceraCompra se inicializa
    CabeceraCompra.TipoDoc=Pedido
    CabeceraCompra.NoSeries=DEVOLU
    CabeceraCompra.No=''
    Se inserta registro CabeceraCompra
    CabeceraCompra.Proveedor=LineasTransfer.DetalleDestino
    CabeceraCompra.Almacen=DEVOLPROV
    Se modifica registro CabeceraCompra
Linea=0
LineasCompra se inicializa
LineasCompra se filtra TipoDoc sea CabeceraCompra.TipoDoc
LineasCompra se filtra NoDoc sea CabeceraCompra.No
Si encuentra ultimo registro LineasCompra entonces
      Linea=LineasCompras.NoLinea
Linea=Linea+1000
LineasCompra se inicializa
LineasCompra.TipoDoc=CabeceraCompra.TipoDoc
LineasCompra.NoDoc=CabeceraCompra.No
LineasCompra.NoLinea=Linea
Se inserta registro LineasCompra
LineasCompra.Tipo=Producto
LineasCompra.No=LineasTransfer.NoProducto
LineasCompra.Almacen=DEVOLPROV
LineasCompra.Cantidad=-(LineasTransfer.CantidadRecibir)
Modificar registro LineasCompra
```

Código Original

```
rcdCabCompra.RESET;
rcdCabCompra.SETCURRENTKEY("Document Type","Buy-from Vendor No.","No.");
rcdCabCompra.SETRANGE("Document Type",rcdCabCompra."Document Type"::Order);
rcdCabCompra.SETRANGE("Buy-from Vendor No.",PARLinTransfer."Detalle Destino");
rcdCabCompra.SETRANGE(Status,rcdCabCompra.Status::Open);
rcdCabCompra.SETFILTER("No.",'DEV-*');

IF NOT rcdCabCompra.FINDFIRST THEN
BEGIN
```

```
  // Creamos un pedido de compra para este proveedor.
  rcdCabCompra.INIT;
  rcdCabCompra."Document Type":=rcdCabCompra."Document Type"::Order;
  rcdCabCompra."No. Series":='DEVOLU';
  rcdCabCompra."No.":='';
  rcdCabCompra.INSERT(TRUE);
  rcdCabCompra.VALIDATE("Buy-from Vendor No.",PARLinTransfer."Detalle Destino");
  rcdCabCompra.VALIDATE(rcdCabCompra."Location Code",'DEVOLPROV');
  rcdCabCompra.MODIFY;
END;

nvNewLin:=0;
rcdLinCompra.RESET;
rcdLinCompra.SETRANGE("Document Type",rcdCabCompra."Document Type");
rcdLinCompra.SETRANGE("Document No.",rcdCabCompra."No.");
IF rcdLinCompra.FINDLAST THEN
  nvNewLin:=rcdLinCompra."Line No.";

nvNewLin+=1000;

// Insertamos la nueva línea
rcdLinCompra.INIT;
rcdLinCompra."Document Type":=rcdCabCompra."Document Type";
rcdLinCompra."Document No.":=rcdCabCompra."No.";
rcdLinCompra."Line No.":=nvNewLin;
rcdLinCompra.INSERT;

rcdLinCompra.Type:=rcdLinCompra.Type::Item;
rcdLinCompra.VALIDATE("No.",PARLinTransfer."Item No.");
rcdLinCompra.VALIDATE("Location Code",'DEVOLPROV');
rcdLinCompra.VALIDATE(Quantity, -PARLinTransfer."Qty. to Receive");

rcdLinCompra."Return Shipment No.":=PARLinTransfer."Document No.";
rcdLinCompra."Return Shipment Line No.":=PARLinTransfer."Line No.";
rcdLinCompra.MODIFY;
```

4.2.2 Compras

En este apartado se explica cómo se generan Pedidos de Compra y Productos.

En lo referente a los Pedidos pueden ser de tres tipos:

- Pedido Central: En este caso, el pedido se recepciona en el almacén CENTRAL y se gestiona desde allí. De este tipo de Pedido su explicación se ha hecho en el apartado Almacén cuando se crea una recepción.
- Pedido Proveedor: Son pedidos que se han recepcionado directamente en la tienda, sin tener que pasar por CENTRAL. En este caso, el personal de compras crea un nuevo pedido de compras, donde debe definir el proveedor correspondiente a ese albarán y en el campo Almacén se debe seleccionar a la

tienda que se le va imputar el pedido. Una vez configurada la cabecera del pedido, se deben ir insertando los productos que vienen indicados en el albarán e indicar sus cantidades correspondientes. Una vez insertados todos los productos, se debe validar el pedido para que el sistema compruebe que en el mes en que se quiere imputar el pago del pedido se tiene prespuesto suficiente para acometer el pago. En caso de no haber presupuesto, se emite un mensaje donde indica ese hecho para que cambien las condiciones de pago. Una vez validado, lo siguiente es registrar el pedido para cargar esos productos a la tienda y generar un movimiento de pago, para que cuando llegue el momento de pagar, se genere un asiento que certifique que se ha realizado el pago de la factura

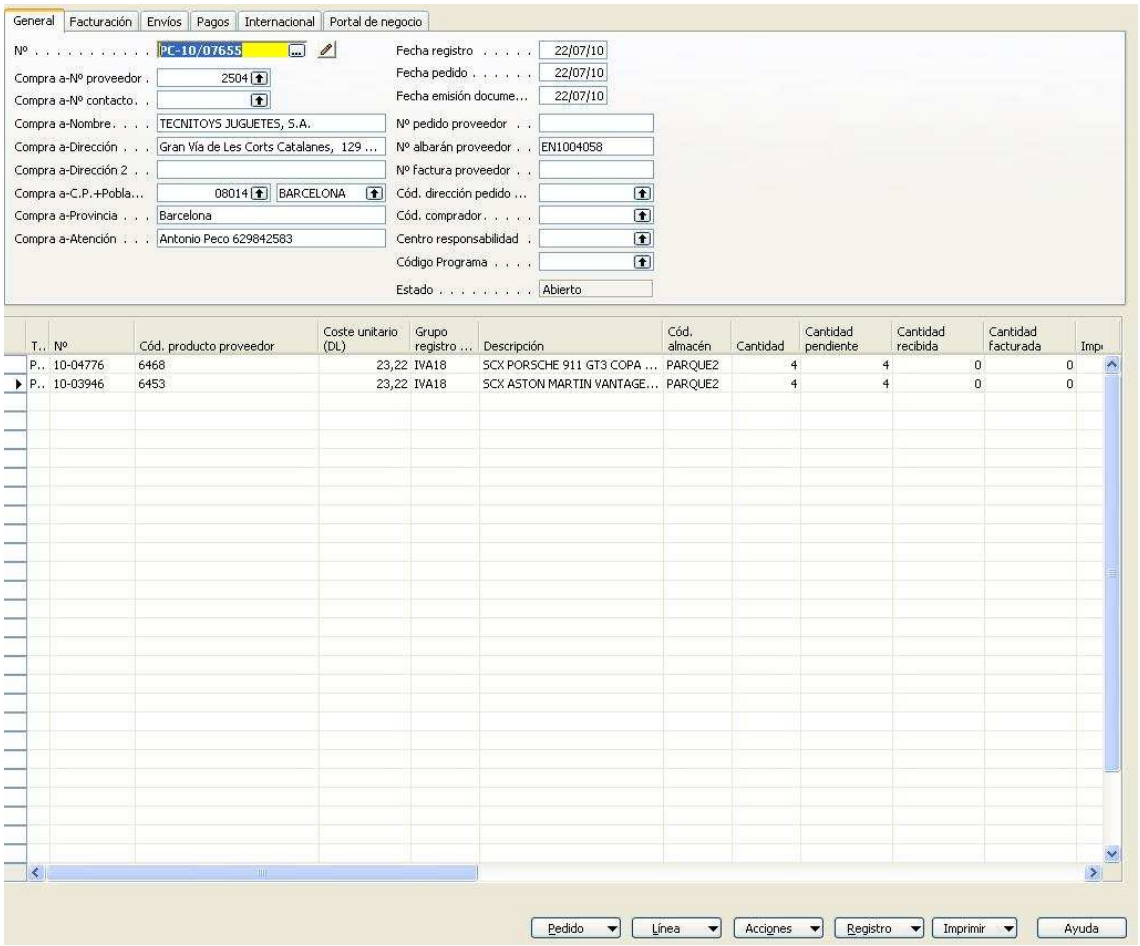


Ilustración 33 Pedido Directo Tienda

- Pedido A Cliente: En este caso es análogo al pedido de proveedor pero con algunas salvedades. En el caso del almacén, se debe poner FRANQUICIAS, para que se impute a ese almacén. Por otro lado, en el campo NoCliente se debe indicar a que cliente va dirigido el pedido. A la hora de facturar, el sistema crea

un pedido de venta adherido a dicho pedido de compra y además, el sistema obliga primero a facturar el pedido de venta antes que el pedido de compra, para que el flujo contable siga un cauce correcto y no haya variaciones contables.

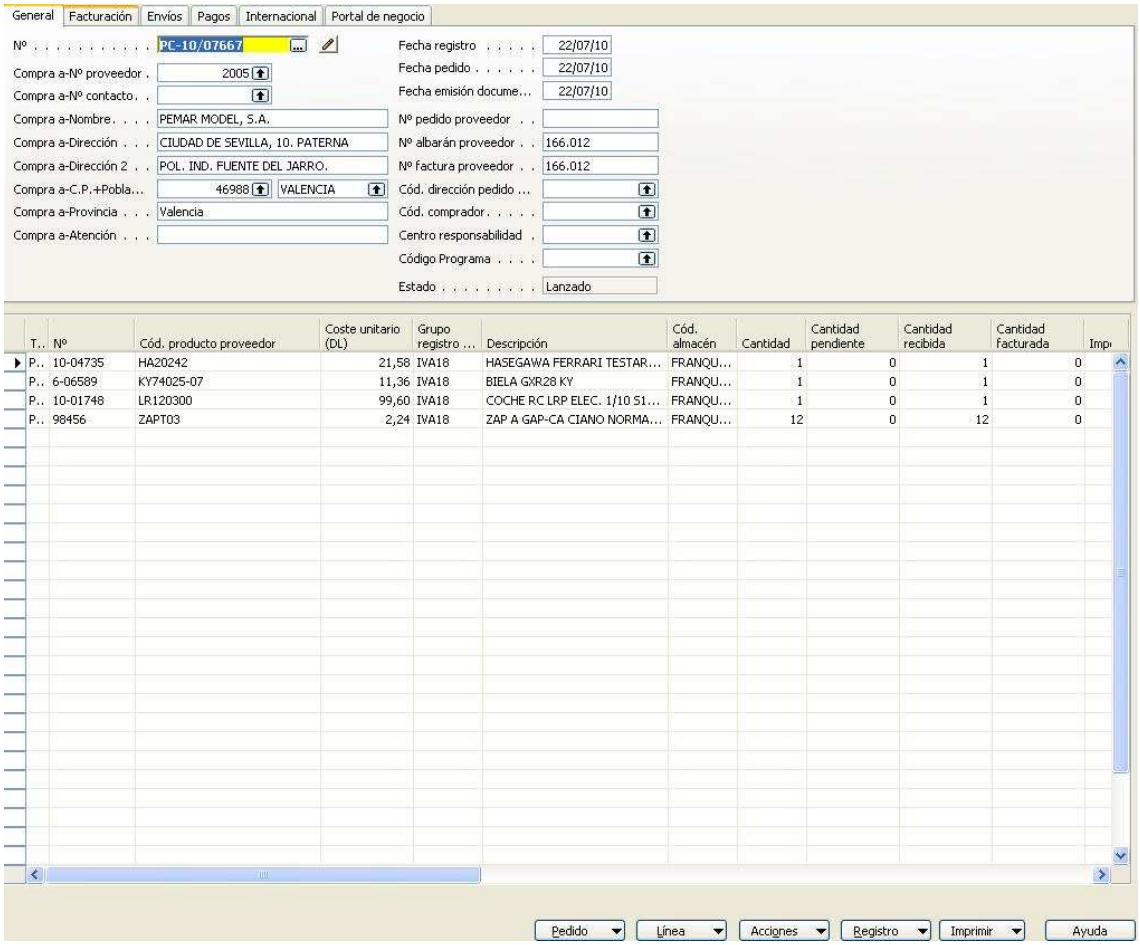


Ilustración 34 Pedido a Cliente

Pedido a Proveedor (Pedido a Central)

ValidarPedido

PseudoCódigo

```
Si Validado entonces
    DesvalidarPedido (‘Pedido desvalidado y vuelto a validar’)
Si CopiaString (No, 1,2) <>’PC’ entonces
    Validado=Verdadero
    Modificar Registro
    Salir de la función
Validado=Falso
Modifica registro
Si Subfamilia=’’ entonces
     ERROR (‘Indicar subfamilia presupuestaria’)
Si TerminosPago=’’ entonces
     ERROR (‘Indicar Condiciones Pago’)
Si FechaVencimiento=’’ entonces
```

```
    ERROR (‘Indicar fecha vencimiento’)
ComprobarPlazos ()
Validado=Verdadero
Modificar Registro
PasaEmpresa (COMPANYNAME)
ImputarPlazos (CabeceraCompra, Verdadero)
HistoricoValidacion se inicializa
HistoricoValidacion.FechaHora=HOY
HistoricoValidacion.NoPedido=No
HistoricoValidacion.Usuario=USERID
HistoricoValidacion.Funcion=Validar
Se inserta registro HistoricoValidacion
```

Código Original

```
IF Validado THEN
BEGIN
  IF NOT CONFIRM ('Este pedido, ya se encuentra validado. De todas formas\'+
            '¿Deseas validarlo de nuevo?') THEN
     ERROR ('Validación cancelada')
  ELSE  //JFG Añadido para que no duplique presupuesto
    DesvalidarPedido ('Pedido desvalidado y vuelto a validar');

END
ELSE
 IF NOT CONFIRM ('¿Estás seguro de ejecutar la validación de este pedido?') THEN
    ERROR ('Proceso cancelado');

IF COPYSTR ("No.", 1, 2) <> 'PC' THEN
BEGIN
 Validado:=TRUE;
 MODIFY;
 MESSAGE ('Pedido validado');
 EXIT;
END;
Validado:=FALSE;
"Cambio Automático de Mes":=FALSE;
MODIFY;
COMMIT;

IF "Shortcut Dimension 1 Code" = '' THEN
  ERROR ('Error. Debes indicar a qué familia presupuestaria será imputado este pedido');

IF "Payment Terms Code" = '' THEN
  ERROR ('Error. Debes indicar el término de pago para poder calcular los vencimientos');

IF "Due Date" = 0D THEN
  ERROR ('Error. No existe fecha de primer vencimiento');

ComprobarPlazos (Rec);
Validado:=TRUE;
MODIFY;
FunPPTO.PasaEmpresa (COMPANYNAME);
FunPPTO.ImputarPlazos (Rec, TRUE);
```

```
rcdHist.RESET;
rcdHist.FechaHora:=CURRENTDATETIME;
rcdHist."No. Pedido":="No.";
rcdHist.Usuario:=USERID;
rcdHist.Funcion:='Validar';
rcdHist.INSERT;
```

DesvalidarPedido (Mensaje)

PseudoCódigo

```
Si Validado entonces
    Validado=Falso
    CambioAutomaticoMes=Falso
    Modificar Registro
    BorrarDetalle (0, No)
    HistoricoValidacion se inicializa
    HistoricoValidacion.FechaHora=HOY
    HistoricoValidacion.NoPedido=No
    HistoricoValidacion.Usuario=USERID
    HistoricoValidacion.Funcion=Desvalidar
    Se inserta registro HistoricoValidacion
```

Código Original

```
IF Validado THEN
BEGIN
  Validado:=FALSE;
  "Cambio Automático de Mes":=FALSE;
  MODIFY;
  FunPPTO.BorraDetalle (0,"No.");
  rcdHist.RESET;
  rcdHist.FechaHora:=CURRENTDATETIME;
  rcdHist."No. Pedido":="No.";
  rcdHist.Usuario:=USERID;
  rcdHist.Funcion:='Desvalidar';
  rcdHist.INSERT;
  MESSAGE (PARMensaje);
END;
```

BorraDetalle (0, No)

PseudoCódigo

```
TablaPresupuestoDetalle se inicializa
TablaPresupuestoDetalle se filtra Tipo sea 0
TablaPresupuestoDetalle se filtra NoDoc sea No
Si encuentra registro TablaPresupuestoDetalle entonces
Repetir
      Borramos registro TablaPresupuestoDetalle
      Si Tipo=Pedido Y PedidoProgramado entonces
            ActualizaPresupuesto (Proveedor, Año, Mes, Familia, Importe)
Hasta registro TablaPresupuestoDetalle sea vacío
```

Código Original

```
PPTODetalle.SETCURRENTKEY(PPTODetalle.Tipo,PPTODetalle."Nº Documento");
```

```
PPTODetalle.SETRANGE(PPTODetalle.Tipo,PARTipo);
PPTODetalle.SETFILTER(PPTODetalle."Nº Documento",PARNDoc);
IF PPTODetalle.FINDSET THEN
REPEAT
  PPTODetalle.DELETE();
  IF (PARTipo=PARTipo::Pedido) AND (PedidoProgramado(PARNDoc)) THEN
     ActualizaPptoProgramado(PPTODetalle."Cód. Proveedor",PPTODetalle."Nº Ppto
Año",PPTODetalle."Nº Ppto Mes",PPTODetalle."Cód. Familia Presupuestaria",-
PPTODetalle.Importe);
UNTIL PPTODetalle.NEXT=0;
```

ActualizaPresupuesto (Proveedor, Año, Mes, Familia, Importe)

PseudoCódigo

```
Tabla PresupuestoDetalle se inicializa
TablaPresupuestoDetalle  se filtra PPtoAño sea Año
TablaPresupuestoDetalle se filtra PPtoMes sea Mes
TablaPresupuestoDetalle se filtra CodFamilia sea Familia
TablaPresupuestoDetalle se filtra Tipo sea Reservado
TablaPresupuestoDetalle se filtra CodProveedor sea Proveedor
Si no encuentra registro TablaPresupuestoDetalle entonces
        Error (‘Presupuesto no Encontrado’);
TablaPresupuestoDetalle.Importe= TablaPresupuestoDetalle.Importe-Importe
Se modifica registro
```

Código Original

```
rcdDetalle.RESET;
rcdDetalle.SETCURRENTKEY("Nº Ppto Año","Nº Ppto Mes","Cód. Familia
Presupuestaria",Tipo,
                    "Fecha Vencimiento","Nº Documento","Cód. Proveedor");
rcdDetalle.SETRANGE("Nº Ppto Año",PARPpto);
rcdDetalle.SETRANGE("Nº Ppto Mes",PARMes);
rcdDetalle.SETRANGE("Cód. Familia Presupuestaria",PARFamilia);
rcdDetalle.SETRANGE(Tipo,rcdDetalle.Tipo::Reservado);
rcdDetalle.SETRANGE("Cód. Proveedor",PARProveedor);
IF NOT rcdDetalle.FINDFIRST THEN
   ERROR('ERROR. No se encuentra el detalle del presupuesto programado para %1, %2, %3,
%4',
        PARProveedor,PARPpto,PARMes,PARFamilia);
rcdDetalle.Importe-=PARImporte;
rcdDetalle.MODIFY;
```

PasaEmpresa (COMPANYNAME)

PseudoCódigo

```
TablaEmpresa se inicializa
TablaEmpresa se filtra Empresa sea COMPANYNAME
```

Código Original

```
rcdEmpresa.GET (PAREmpresa);
```

ImputarPlazos

PseudoCódigo

```
ImportePendiente=TablaPedido.ImporteNoFacturado
Si ImportePendiente=0 entonces
    Salir
TerminosPago.CambioEmpresa (TablaEmpresa.Name)
Si no TerminosPago.Coge (TablaPedido.CondicionesPago) entonces
      Error (‘Pedido con termino de pago desconocido’)
dvFechaVcto=TablaPedido.FechaVencimiento
TablaPlazos se inicializa
TablaPlazos.CambioEmpresa (TablaEmpresa.Name)
TablaPlazos se filtra CodigoTerminosPago sea TerminosPago.Code
Si no encuentra registros TablaPlazos entonces
     GrabaDetalle (TablaPedido.DimensionSubFamilia, dvFechaVcto,ImportePendiente,
Pedido,TablaPedido.No,’’,TablaPedido.Proveedor,TablaPedido.FechaDocumento,
TablaPedido.CodigoDepartamento,TablaPedido.CambioAutomatico, TablaPedido.Subfamilia)
    SumaImporte (TablaPedido.DimensionSubfamilia, dvFechaVcto, ImportePendiente,
Pedido)
Sino
     Si TerminosPago.DistribucionIVA<>Proporcional entonces
          Error (‘Opcion no contemplada’);
     Repetir
         ImportePlazo=ImportePendiente+TablaPlazos.%delTotal/100;
         Si InicioGlobal=’’ y FinGlobal=’’ entonces
            FamiliaPresupuesto se inicializa
            FamiliaPresupuesto se filtra CodFamilia sea TablaPedido.DimSubfamilia
            Si encuentra registro FamiliaPresupuesto entonces
            Repetir
                PresupuestoAnual se inicializa
                PrresupuestoAnual.Coge (FamiliaPresupuesto.PptoAño)
                Si PresupuestoAnual.Activo entonces
                   Si InicioGlobal=’’ O InicioGlobal>PresupuestoAnual.FechaIni
                       InicioGlobal=PresupuestoAnual.FechaIni
                       Si FinGlobal=’’ O FinGlobal<PresupuestoAnual.FechaFin
                            FinGlobal=PresupuestoAnual.FechaFin
            Hasta registro FamiliaPresupuesto sea vacío
            Si dvFechaVcto>=InicioGlobal Y dvFechaVcto<=Fin Global
                GrabaDetalle (TablaPedido.DimensionSubfamilia, dvFechaVcto, ImportePlazo,
Pedido, TablaPedido.Proveedor, TablaPedido.FechaDocumento,
TablaPedido.CodigoDepartamento, TablaPedido.CambioAutomatico,
TablaPedido.CodigoSubfamilia)
                SumaImporte (TablaPedido.DimensionSubfamilia, dvFechaVcto, ImportePlazo,
Pedido)
                dvFechaVcto=Calcular(TablaPlazos.Fecha)
                AjusFechVcto(dvFechaVcto)
     Hasta registro TablaPlazos sea vacío
```

Código Original

```
nvImportePendiente:=rcdPedido.ImporteNoFacturado(rcdPedido);
IF nvImportePendiente = 0 THEN EXIT;

TerminosPago.CHANGECOMPANY(rcdEmpresa.Name);
```

```
IF NOT TerminosPago.GET(rcdPedido."Payment Terms Code") THEN
  ERROR('Pedido con término de pago desconocido.\'+
       'Pedido #1############\'+
       'Empresa #2############################\'+
        'NO SE DÓNDE IMPUTAR EL PAGO',rcdPedido."No.",rcdEmpresa.Name);

dvFechaVcto:=rcdPedido."Due Date";

rcdPlazos.RESET;
rcdPlazos.CHANGECOMPANY(rcdEmpresa.Name);
rcdPlazos.SETRANGE("Payment Terms Code",TerminosPago.Code);
IF NOT rcdPlazos.FINDSET() THEN
BEGIN
   GrabaDetalle(DevDimValorPresup(rcdPedido."Shortcut Dimension 1 Code"),
             dvFechaVcto,nvImportePendiente,optTipo::Pedido,rcdPedido."No.",",
             rcdPedido."Buy-from Vendor No.",rcdPedido."Document Date",rcdPedido."Shortcut
Dimension 2 Code",
             rcdPedido."Cambio Automático de Mes",rcdPedido."Shortcut Dimension 1 Code");
   SumaImporte(DevDimValorPresup(rcdPedido."Shortcut Dimension 1
Code"),dvFechaVcto,nvImportePendiente,optTipo::Pedido);
END
ELSE
BEGIN
 IF TerminosPago."VAT distribution" <> TerminosPago."VAT distribution"::Proportional
THEN
 BEGIN
    ERROR('No se encuentra contemplada la distribución de IVA no proporcional.\'+
        'Pedido #1############\'+
        'Empresa #2############################',rcdPedido."No.",rcdEmpresa.Name);
 END;
 REPEAT
  rcdPlazos.TESTFIELD("% of Total");
  nvImportePlazo :=  nvImportePendiente * rcdPlazos."% of Total" / 100;

  IF ((dvInicioGlobal=0D) AND (dvFinGlobal=0D)) THEN BEGIN
    FamiliaPresup.SETRANGE(FamiliaPresup."Cód. Familia
Presupuestaria",DevDimValorPresup(rcdPedido."Shortcut Dimension 1 Code"));
    IF FamiliaPresup.FIND('-') THEN REPEAT
      AnualPresup.GET(FamiliaPresup."Nº Ppto Año");
      IF AnualPresup.Activo THEN BEGIN
        IF (dvInicioGlobal=0D) OR (dvInicioGlobal>AnualPresup."Fecha Inicial") THEN
          dvInicioGlobal:=AnualPresup."Fecha Inicial";
        IF (dvFinGlobal=0D) OR (dvFinGlobal<AnualPresup."fecha final")THEN
          dvFinGlobal:=AnualPresup."fecha final";
      END;
    UNTIL FamiliaPresup.NEXT=0;
  END;

  IF (dvFechaVcto >= dvInicioGlobal) AND (dvFechaVcto <= dvFinGlobal) THEN
  BEGIN
    GrabaDetalle(DevDimValorPresup(rcdPedido."Shortcut Dimension 1 Code"),
           dvFechaVcto,nvImportePlazo,optTipo::Pedido,rcdPedido."No.",",
           rcdPedido."Buy-from Vendor No.",rcdPedido."Document
Date",rcdPedido."Shortcut Dimension 2 Code",
```

```
            rcdPedido."Cambio Automático de Mes",rcdPedido."Shortcut Dimension 1 Code");
    SumaImporte(DevDimValorPresup(rcdPedido."Shortcut Dimension 1
Code"),dvFechaVcto,nvImportePlazo,optTipo::Pedido);
  END;

  dvFechaVcto := CALCDATE(rcdPlazos."Gap between Installments",dvFechaVcto);
  AjusFechVto.PurchAdjustDueDate (dvFechaVcto, rcdPedido."Pay-to Vendor No.");

 UNTIL rcdPlazos.NEXT=0;
END;
```

GrabaDetalle (TablaPedido.DimensionSubfamilia, dvFechaVcto, ImportePlazo, Pedido, TablaPedido.Proveedor, TablaPedido.FechaDocumento, TablaPedido.CodigoDepartamento, TablaPedido.CambioAutomatico, TablaPedido.CodigoSubfamilia)

PseudoCódigo

```
ConfiguracionEmpresa.Coger
TablaDimension.Coger (ConfiguracionEmpresa.CodigoSubfamilia,
TablaPedido.DimensionSubfamilia)
ConfiguracionCompras.Coger;
Si ConfiguraciónCompras.SegundoControlPpto entonces
     SegundoControl (TablaPedido.DimensionSubfamilia, dvFechaVcto, ImportePlazo, Pedido,
TablaPedido.Proveedor, TablaPedido.FechaDocumento, TablaPedido.CodigoDepartamento,
TablaPedido.CambioAutomatico, TablaPedido.CodigoSubfamilia)
TablaPresupuesto se filtra Activo sea Verdadero
TablaPresupuesto se filtra Tipo Presupuesto sea TablaDimension.TipoPresupuesto
Si no encuentra registro TablaPresupuesto entonces
    ERROR (‘No existe presupuesto activo’);
Repetir
   PresupuestoMes se inicializa
   PresupuestoMes se filtra PptoAño sea TablaPresupuesto.PptoAño
   Si encuentra registro PresupuestoMes entonces
      Repetir
         Si dvFechaVcto>=PresupuestoMes.FechaInicial y dvFechaVcto<=
         PresupuestoMes.FechaFin entonces
              TablaFamilia se inicializa
              TablaFamilia se filtra PptoAño sea TablaPresupuesto.PptoAño
              TablaFamilia se filtra PptoMes sea PresupuestoMes.PptoMes
              TablaFamilia se filtra CodigoFamilia sea TablaPedido.DimSubfamilia
              Si no encuentr registro TablaFamilia entonces
                   Error (‘No se encuentra el detalle’);
              numLinea=1
             TablaDetalle se inicializa
             TablaDetalle buscar ultimo registro
             Si encuentra registro TablaDetalle entonces
                   numLinea=TablaDetalle.NoLinea+1
             TablaDetalle se inicializa
             TablaDetalle.NoLinea=numLinea
             TablaDetalle.PptoAño=TablaPresupuesto.PptoAño
             TablaDetalle.PptoMes=PresupuestoMes.PptoMes
             TablaDetalle.CodFamilia= TablaPedido.DimensionSubfamilia
```

```
                TablaDetalle.Tipo=Pedido
                TablaDetalle.FechaVencimiento=dvFechVcto
                TablaDetalle.NºDocumento=TablaPedido.NoDoc
                TablaDetalle.NºEfecto=TablaPedido.NoEfecto
                TablaDetalle.Importe=ImportePlazo
                Si TablaEmpresa.Nombre=’’ entonces
                    TablaDetalle.Empresa=COMPANYNAME
               Sino
                    TablaDetalle.Empresa=TablaEmpresa.Nombre
               TablaDetalle.CodProveedor= TablaPedido.Proveedor
               TablaDetalle.FechaPedido= TablaPedido.FechaDocumento
               TablaDetalle.CodDepartamento=TablaPedido.CodigoDepartamento
               TablaDetalle.CambioAutomatico=TablaPedido.CambioAutomatico
               TablaDetalle.CodSubfamilia= TablaPedido.CodigoSubfamilia
               TablaDetalle.FechaCreacion=TODAY
               TablaDetalle.HoraCreacion=TIME
               TablaDetalle.IdUsuario=USERID
               Insertar Registro TablaDetalle
               Si Tipo=Pedido Y PedidoProgramado entonces
                   TablaDetalle.PedidoPlantilla=Verdadero
                   TablaDetalle se modifica
                   ActualizarPresupuesto
               SALIR
        Hasta registro PresupuestoMes sea vacío
        SALIR
Hasta registro TablaPresupuesto sea vacío
ERROR (‘ No encuentra detalle’)
```

Código Original

```
GLSetup.GET;
DimVal.GET(GLSetup."Global Dimension 1 Code",PARFamilia);

PurchSetup.GET;
IF PurchSetup."Segundo Control Ppto. Activo" THEN

SegundoControl(PARFamilia,PARVcto,PARImporte,PARTipo,PARDocumento,PAREfecto,P
ARProveedor,PARFechaPed,
            PARDpto,PARCambiaVcto,PARSubfamilia);

// Primero buscamos el mes al que corresponde el vencimiento
rcdPresup.SETRANGE(Activo,TRUE);

rcdPresup.SETRANGE(rcdPresup."Tipo Presupuesto",DimVal."Tipo Presupuesto");
IF NOT rcdPresup.FINDSET() THEN
  ERROR('No existe ningún presupuesto activo');
REPEAT
 rcdMes.RESET;
 rcdMes.SETRANGE("Nº Ppto Año",rcdPresup."Nº Ppto Año");

 IF rcdMes.FINDSET() THEN
 BEGIN
  REPEAT
   IF (PARVcto >= rcdMes."Fecha Inicial") AND (PARVcto <= rcdMes."fecha final") THEN
   BEGIN
```

```
    rcdFamilia.RESET;
    rcdFamilia.SETRANGE("Nº Ppto Año",rcdPresup."Nº Ppto Año");
    rcdFamilia.SETRANGE("Nº Ppto Mes",rcdMes."Nº Ppto Mes");
    rcdFamilia.SETRANGE("Cód. Familia Presupuestaria",PARFamilia);
    IF NOT rcdFamilia.FINDFIRST() THEN
      ERROR('Error. No se encuentra detalle para la clave %1 - %2 - %3 - %4'
          ,FORMAT(rcdPresup."Nº Ppto Año"),FORMAT(rcdMes."Nº Ppto
Mes"),FORMAT(PARFamilia),PARDocumento);

    numLinea:=1;
    rcdDetalle.RESET;
    IF rcdDetalle.FINDLAST() THEN
      numLinea:=rcdDetalle."Nº Linea"+1;

    rcdDetalle.INIT;
    rcdDetalle."Nº Linea":=numLinea;
    rcdDetalle."Nº Ppto Año":=rcdPresup."Nº Ppto Año";
    rcdDetalle."Nº Ppto Mes":=rcdMes."Nº Ppto Mes";
    rcdDetalle."Cód. Familia Presupuestaria":=PARFamilia;
    rcdDetalle.Tipo:=PARTipo;
    rcdDetalle."Fecha Vencimiento":=PARVcto;
    rcdDetalle."Nº Documento":=PARDocumento;
    rcdDetalle."Nº Efecto":=PAREfecto;
    rcdDetalle.Importe:=PARImporte;
    IF rcdEmpresa.Name = '' THEN
       rcdDetalle.Empresa:=COMPANYNAME
    ELSE
      rcdDetalle.Empresa:=rcdEmpresa.Name;
    rcdDetalle."Cód. Proveedor":=PARProveedor;
    rcdDetalle."Fecha Pedido":=PARFechaPed;
    rcdDetalle."Cód. Departamento":=PARDpto;
    rcdDetalle."Cambio Automático de Mes":=PARCambiaVcto;
    rcdDetalle."Cód. Subfamilia":=PARSubfamilia;
    rcdDetalle."Fecha Creación":=TODAY;
    rcdDetalle."Hora Creación":=TIME;
    rcdDetalle."Id. Usuario":=USERID;
    rcdDetalle.INSERT;

    // Si es un pedido 'programado', hay que reducir el presupuesto disponible para ese
proveedor-familia-mes
    IF (PARTipo=PARTipo::Pedido) AND (PedidoProgramado(PARDocumento)) THEN
    BEGIN
      rcdDetalle."Pedido De Plantilla":=TRUE;
      rcdDetalle.MODIFY;
      ActualizaPptoProgramado(PARProveedor,rcdPresup."Nº Ppto Año",rcdMes."Nº Ppto
Mes",PARFamilia,PARImporte);
    END;
    EXIT;
   END;
  UNTIL rcdMes.NEXT=0;
 END;
UNTIL rcdPresup.NEXT=0;
// Si llega hasta aquí es que hay algún error.
ERROR('Error. No se encuentra detalle para la clave %1 - %2 - %3 - %4'
```

```
    ,FORMAT(rcdPresup."Nº Ppto
Año"),FORMAT(PARVcto),FORMAT(PARFamilia),PARDocumento);
```

SumaImporte (TablaPedido.DimensionSubfamilia, dvFechaVcto,

ImportePendiente, Pedido)

PseudoCódigo

```
ConfiguracionEmpresa.Coger
TablaDimension.Coger (ConfigurarEmpresa.CodSubfamilia,
                      TablaPedido.DimensionSubfamilia)
TablaPrespuesto se filtra Activo sea verdadero
TablaPresupuesto se filtra TipoPresupuesto sea TablaDimension.TipoPresupuesto
Si no encuentra registro TablaPresupuesto entonces
     Error (‘No existe presupuesto Activo’)
Repetir
    PresupuestoMes se inicializa
    PresupuestoMes se filtra PptoAño sea TablaPresupuesto.PptoAño
    Si encuentra registros PresupuestoMes entonces
       Repetir
          Si dvFechaVcto>=PresupuestoMes.FechaIni Y dvFechaVcto<=
          PresupuestoMes.FechaFin entonces
               TablaFamilia se inicializa
             TablaFamilia se filtra PptoAño sea TablaPresupuesto.PptoAño
             TablaFamilia se filtra PptoMes sea PresupuestoMes.PptoMes
             TablaFamilia se filtra CodFamilia sea TablaPedido.DimensionSubfamilia
             Si no encuentra registros TablaFamilia entonces
                   Error (‘No se encuentra el detalle’)
            Caso Tipo
                   Pedido: TablaFamilia.ImportePedido+=ImportePendiente
                   Cartera: TablaFamilia.ImporterCartera+=ImportePendiente
                   Pagado: TablaFamilia.ImportePagado+=ImportePendiente
            TablaFamilia.ImporteTotal= TablaFamilia.ImportePedido+
            TablaFamilia.ImporterCartera+ TablaFamilia.ImportePagado
            Modificar Registro
            Salir
       Hasta registro PresupuestoMes sea vacío
Hasta registro TablaPresupuesto sea vacío
Error (‘No se encuentra detalle’);
```

Código Original

```
GLSetup.GET;
DimVal.GET(GLSetup."Global Dimension 1 Code",PARFamilia);
rcdPresup.SETRANGE(Activo,TRUE);
rcdPresup.SETRANGE(rcdPresup."Tipo Presupuesto",DimVal."Tipo Presupuesto");
IF NOT rcdPresup.FINDSET() THEN
  ERROR('No existe ningún presupuesto activo');
REPEAT
 rcdMes.RESET;
 rcdMes.SETRANGE("Nº Ppto Año",rcdPresup."Nº Ppto Año");
 IF rcdMes.FINDSET() THEN
 BEGIN
  REPEAT
   IF (PARVcto >= rcdMes."Fecha Inicial") AND (PARVcto <= rcdMes."fecha final") THEN
```

```
   BEGIN
     rcdFamilia.RESET;
     rcdFamilia.SETRANGE("Nº Ppto Año",rcdPresup."Nº Ppto Año");
     rcdFamilia.SETRANGE("Nº Ppto Mes",rcdMes."Nº Ppto Mes");
     rcdFamilia.SETRANGE("Cód. Familia Presupuestaria",PARFamilia);
     IF NOT rcdFamilia.FINDFIRST() THEN
       ERROR('Error. No se encuentra detalle para la clave %1 - %2 - %3'
           ,FORMAT(rcdPresup."Nº Ppto Año"),FORMAT(rcdMes."Nº Ppto
Mes"),FORMAT(PARFamilia));

     CASE PARTipo OF
      PARTipo::Pedido: rcdFamilia."Importe Consumido Pedido" += PARImporte;
      PARTipo::Cartera: rcdFamilia."Importe Consumido Cartera"+= PARImporte;
      PARTipo::Pagado: rcdFamilia."Importe Consumido Pagado"+= PARImporte;
     END;

     rcdFamilia."Importe Consumido Total" := rcdFamilia."Importe Consumido Pedido" +
rcdFamilia."Importe Consumido Cartera"+
                              rcdFamilia."Importe Consumido Pagado";
     rcdFamilia.MODIFY;
     EXIT;
    END;
   UNTIL rcdMes.NEXT=0;
  END;
UNTIL rcdPresup.NEXT=0;
// Si llega hasta aquí es que hay algún error.
ERROR('Error. No se encuentra detalle para la clave %1 - %2 - %3'
     ,FORMAT(rcdPresup."Nº Ppto Año"),FORMAT(PARVcto),FORMAT(PARFamilia));
```

SegundoControl (TablaPedido.DimensionSubfamilia, dvFechaVcto, ImportePlazo, Pedido, TablaPedido.Proveedor, TablaPedido.FechaDocumento, TablaPedido.CodigoDepartamento, TablaPedido.CambioAutomatico, TablaPedido.CodigoSubfamilia)

PseudoCódigo

```
ConfiguracionEmpresa.Coger
TablaDimension.Coger (ConfiguracionEmpresa.Subfamilia,
                              TablaPedido.DimensionSubfamilia)
PresupuestoMes se filtra TipoPresupuesto sea TablaDimension.TipoPresupuesto
PresupuestoMes se filtra FechaInicial sea menor o igual dvFechVcto
PresupuestoMes se filtra FechaFinal sea mayor o igual dvFechVcto
Si no encuentra registros PresupuestoMes entonces
       Error (‘No existe presupuesto’);
TablaFamilia se filtra PptoAño sea PresupuestoMes.PptoAño
TablaFamilia se filtra PptoMes sea PresupuestoMes.PptoMes
TablaFamilia se filtra CodFamilia sea TablaPedido.DimensionSubfamilia
Si no encuentra registro TablaFamilia entonces
       Mensaje (‘No existe presupuesto creado’);
Si tipo=Pedido Y PedidoProgramado entonces
     TablaFamilia se filtra FiltroProveedor sea TablaPedido.Proveedor
     TablaFamilia se calcula ImporteReservado
     Si Tablafamilia.ImporterReservado<ImportePlazo entonces
        ERROR (‘Presupuesto Sobrepasado):
     Sino
```

TablaFamilia se calcula ImporteConsumido
Si TablaFamilia.ImporteConsumido+(-1*ImportePlazo)>
TablaFamilia.ImportePresupuestado entonces
ERROR (‘Presupuesto Sobrepasado’)

Código Original

```
GLSetup.GET;
DimVal.GET(GLSetup."Global Dimension 1 Code",PARFamilia);

rcdPresupMes.SETRANGE(rcdPresupMes."Tipo Presupuesto",DimVal."Tipo Presupuesto");
rcdPresupMes.SETRANGE("Fecha Inicial",0D,PARVcto);
rcdPresupMes.SETRANGE("fecha final",PARVcto,31129999D);

IF NOT rcdPresupMes.FINDFIRST() THEN
  ERROR('No existe ningún presupuesto creado para %1',FORMAT(PARVcto));

rcdPresupFamilia.SETRANGE(rcdPresupFamilia."Tipo Presupuesto",DimVal."Tipo
Presupuesto");
rcdPresupFamilia.SETRANGE("Nº Ppto Año",rcdPresupMes."Nº Ppto Año");
rcdPresupFamilia.SETRANGE("Nº Ppto Mes",rcdPresupMes."Nº Ppto Mes");
rcdPresupFamilia.SETRANGE("Cód. Familia
Presupuestaria",FunPPTO.DevDimValorPresup(PARFamilia));
IF NOT rcdPresupFamilia.FINDFIRST() THEN
  MESSAGE('No existe ningún presupuesto creado para %1 - %2 - %3',
       FORMAT(rcdPresupMes."Nº Ppto Año"),FORMAT(rcdPresupMes."Nº Ppto Mes"),
       FunPPTO.DevDimValorPresup(PARFamilia));
IF (PARTipo=PARTipo::Pedido) AND (PedidoProgramado(PARDocumento)) THEN
BEGIN
   rcdPresupFamilia.SETFILTER("Filtro Proveedor",PARProveedor);
   rcdPresupFamilia.CALCFIELDS("Importe Reservado Pendiente");
   IF rcdPresupFamilia."Importe Reservado Pendiente" < PARImporte THEN
     ERROR('¡¡¡¡¡ ATENCION !!!!!!!\\'+
         'PRESUPUESTO PROGRAMADO PARA '+FORMAT(PARProveedor) +'
SOBREPASADO\'+
         '-------------------------------------------------------\'+
         '\Pedido ' + FORMAT(PARDocumento) +
         '\Presupuesto ' + FORMAT(rcdPresupFamilia."Nº Ppto Año") +
         '\Mes ' + FORMAT(rcdPresupMes."Descripción Periodo") +
         '\Familia ' + FORMAT(rcdPresupFamilia."Cód. Familia Presupuestaria") +
         '\Saldo Disponible ' + FORMAT(rcdPresupFamilia."Importe Reservado Pendiente") +
         '\Importe Vencimiento ' + FORMAT(PARImporte));
END
ELSE
BEGIN
 rcdPresupFamilia.CALCFIELDS(rcdPresupFamilia."Importe Consumido Total");
 IF rcdPresupFamilia."Importe Consumido Total"+(-1*PARImporte) >
rcdPresupFamilia."Importe Presupuestado" THEN
   ERROR('¡¡¡¡¡ ATENCION !!!!!!!\\'+
       'SEGUNDO CONTROL.\'+
       'PRESUPUESTO SOBREPASADO\'+
       '--------------------------------------\'+
       '\Documento ' + FORMAT(PARDocumento) + 'Efecto '+FORMAT(PAREfecto)+
       '\Presupuesto ' + FORMAT(rcdPresupMes."Nº Ppto Año") +
```

```
        '\Mes ' + FORMAT(rcdPresupMes."Descripción Periodo") +
        '\Familia ' + FORMAT(rcdPresupFamilia."Cód. Familia Presupuestaria") +
        '\Importe Presupuestado ' + FORMAT(rcdPresupFamilia."Importe Presupuestado") +
        '\Importe Consumido ' + FORMAT(rcdPresupFamilia."Importe Consumido Total") +
        '\Importe Vencimiento ' + FORMAT(-1*PARImporte));
END;
```

Pédido a Cliente

En este caso, las funciones que utiliza son las mismas que los otros dos pedidos a excepción de una función que sólo se ejecuta en caso de ser pedido a cliente.

ComprobarEnvioDirecto (CabeceraCompra)

PseudoCódigo

```
Si CabeceraCompra.Cliente=’’ entonces
   Salir;
LineasCompras se inicializa
LineasCompras se filtra TipoDocumento sea CabeceraCompra.TipoDocumento
LineasCompras se filtra NoDocumento sea CabeceraCompra.No
Si no encuentra registro LineasCompras entonces
      ERROR (‘No existe líneas en el pedido);
bEnvioDirecto=Falso
bNoEnvioDirecto=Falso
Repetir
    Si LineasCompras.Almacen=CENTRAL entonces
        ERROR (‘Almacen Incorrecto);
        Si LineasCompras.EnvioDirecto entonces
            bEnvioDirecto=Verdadero
        Sino
            bNoEnvioDirecto=Verdadero
Hasta registro LineasCompras sea vacío
Si bEnvioDirecto Y bNoEnvioDirecto entonces
      Error (‘Hay líneas con envio directo y líneas sin envio directo):
Si bEnvioDirecto entonces
      EXIT;
CabeceraVenta se inicializa
CabeceraVenta.TipoDocumento=Pedido
CabeceraVenta.FechaRegistro=CabeceraCompra.FechaRegistro
CabeceraVenta.FechaDocumento=CabeceraCompra.FechaDocumento
CabeceraVenta.Cliente=CabeceraCompra.Cliente
Insertar Registro CabeceraVenta
CabeceraVenta.Almacen=CabeceraCompra.Almacen
Modificar registro CabeceraVenta
CabeceraCompra se posiciona primer registro
Repetir
    LineasVenta se inicializa
    LineasVenta.TipoDocumento=CabeceraVenta.TipoDocumento
    LineasVenta.NoDocumento=CabeceraVenta.No
    LineasVenta.NoLine=LineasCompra.NoLine
    LineasVenta.Tipo=LineasCompra.Tipo
    LineasVenta.Cantidad=LineasCompra.Cantidad
    LineasVenta.CosteUnitario=LineasCompra.CosteUnitario
    LineasVenta.CodigoCompra=ENVDIRECTO
```

```
    LineasVenta.CodigoAlmacen=LineasCompra.CodigoAlmacen
    Si LineasCompra.Tipo=Producto entonces
        LineasVenta.EnvioDirecto=Verdadero
    LineasVenta.NoPedidoCompra=LineasCompra.NoDocumento
    LineasVenta.NoLineaPedidoCompra=LineasCompra.NoLinea
    Si LineasCompra.Tipo=Cuenta entonces
        LineasVenta.PrecioVenta=LineasCompra.CosteUnitario
        Insertar registro LineasVenta
        LineasCompra.EnvioDirecto=Verdadero
        LineasCompra.CodigoCompra=LineasVenta.CodigoCompra
        LineasCompra.NoPedidoVenta=LineasVenta.NoDocumento
        LineasCompra.NoLineaPedidoVenta=LineasVenta.NoLinea
        Modificar registro LineasCompra
Hasta regsitro CabeceraCompra sea vacío
```

Código Original

```
IF PARrcdCabCompra."Sell-to Customer No." = '' THEN
  EXIT;

rcdLinCompra.RESET;
rcdLinCompra.SETRANGE("Document Type",PARrcdCabCompra."Document Type");
rcdLinCompra.SETRANGE("Document No.",PARrcdCabCompra."No.");

IF NOT rcdLinCompra.FINDSET THEN
  ERROR('Error. No existen líneas en este pedido');

bEnvDirecto:=FALSE;
bNoEnvDirecto:=FALSE;
REPEAT
 IF rcdLinCompra."Location Code" = 'CENTRAL' THEN
   ERROR('Error. No puedes registrar un envío directo usando cómo almacén de entradas
CENTRAL');
 IF rcdLinCompra."Drop Shipment" THEN
   bEnvDirecto:=TRUE
 ELSE
   bNoEnvDirecto:=TRUE;
UNTIL rcdLinCompra.NEXT=0;

IF (bEnvDirecto) AND (bNoEnvDirecto) THEN
  ERROR('Error. Tienes líneas ligadas a un pedido de venta, y otras que no');

IF bEnvDirecto THEN
  EXIT;
IF NOT CONFIRM('¡¡¡ ATENCION !!!\'+
          'Has indicado que este pedido sea un pedido de envío directo a %1\' +
          'Se creará un pedido de venta para este cliente.\'+
          '¿Deseas continuar?',FALSE,PARrcdCabCompra."Ship-to Name") THEN
  ERROR('Proceso cancelado');

rcdCabVenta.INIT;
rcdCabVenta."Document Type":=rcdCabVenta."Document Type"::Order;
rcdCabVenta."Posting Date":=PARrcdCabCompra."Posting Date";
rcdCabVenta."Document Date":=PARrcdCabCompra."Document Date";
```

```
rcdCabVenta.VALIDATE("Sell-to Customer No.",PARrcdCabCompra."Sell-to Customer
No.");
rcdCabVenta.INSERT(TRUE);
rcdCabVenta."Location Code":=PARrcdCabCompra."Location Code";
rcdCabVenta.MODIFY;

rcdLinCompra.FINDSET();
REPEAT
  rcdLinVenta.INIT;
  rcdLinVenta."Document Type":=rcdCabVenta."Document Type";
  rcdLinVenta."Document No.":=rcdCabVenta."No.";
  rcdLinVenta."Line No.":=rcdLinCompra."Line No.";
  rcdLinVenta.Type:=rcdLinCompra.Type;
  rcdLinVenta.VALIDATE("No.",rcdLinCompra."No.");
  rcdLinVenta.VALIDATE(Quantity,rcdLinCompra.Quantity);
  rcdLinVenta.VALIDATE("Unit Cost (LCY)",rcdLinCompra."Unit Cost (LCY)");
  rcdLinVenta."Purchasing Code" := 'ENVDIRECTO';
  rcdLinVenta."Location Code":=rcdLinCompra."Location Code";
  //Si es tipo cuenta, luego no te deja facturar el pedido de venta
  IF rcdLinCompra.Type=rcdLinCompra.Type::Item THEN
     rcdLinVenta."Drop Shipment":=TRUE;
  rcdLinVenta."Purchase Order No.":=rcdLinCompra."Document No.";
  rcdLinVenta."Purch. Order Line No.":=rcdLinCompra."Line No.";
  //EN este caso, se debe validar el precio de venta con lo que se ha metido por cuenta
  IF rcdLinCompra.Type=rcdLinCompra.Type::"G/L Account" THEN
    rcdLinVenta.VALIDATE(rcdLinVenta."Unit Price",rcdLinCompra."Unit Cost (LCY)");
  rcdLinVenta.INSERT();
// Ligamos la línea del pedido de compra con la de venta
  rcdLinCompra."Drop Shipment":=TRUE;
  rcdLinCompra."Purchasing Code" := rcdLinVenta."Purchasing Code";
  rcdLinCompra."Sales Order No.":=rcdLinVenta."Document No.";
  rcdLinCompra."Sales Order Line No.":=rcdLinVenta."Line No.";
  rcdLinCompra.MODIFY;
UNTIL rcdLinCompra.NEXT=0;
```

Insertar Productos

En esta funcionalidad, se explica la forma en la que se insertan los productos. En primer lugar, el personal de compras debe rellenar una serie de campos. A partir de esa serie de campos, el sistema va calcualando otros campos. Los campos que inserta el personal son: CodigoProveedor, Subfamilia, Cod. Grupo Producto, Nº Proveedor, CodigoFabricante, PrecioCoste, Tarifa Proveedor, %Beneficio. A partir de esos campos, el sistema rellena otros campos realizando una serie de cálculos.

Insertar Registro

PseudoCódigo

```
Método Coste=Promedio
CalculoPrecio=Coste+beneficio
FechaCreacion=HOY
UsuarioCreacion=IDUSUARIO
```

TablaUnidades se inicializa
TablaUnidades.NoProducto=No
TablaUnidades.CantidadxUnidad=1
Insertar Registro
TablaCodigoBarras se inicializa
TablaCodigoBarras.Producto=No
cBarra=ConvertirString (No,-,/)
TablaCodigoBarras.CodBarrasAsociado=cBarra
TablaCodigoBarras.BarrasP=Verdadero
Insertar Registro

Código Original

```
// Inicicalizamos campos
"Costing Method":="Costing Method"::Average;
"Price/Profit Calculation":="Price/Profit Calculation"::"Price=Cost+Profit";
"Fecha Creación":=TODAY;
"Usuario Creación":=USERID;
// Grabamos la unidad de medida por defecto
rcdUd.INIT;
rcdUd."Item No." := "No.";
rcdUd.Code := 'UD.';
rcdUd."Qty. per Unit of Measure" := 1;
rcdUd.INSERT;
//Grabar Código Barras
rcdBarras.INIT;
rcdBarras.Producto:="No.";
cBarra:=CONVERTSTR("No.",'-','/');
rcdBarras."Cod. barras asociado":=cBarra;
rcdBarras."Barras Poly":=TRUE;
rcdBarras.INSERT;
```

Validar Campos

Calculo Precio/Beneficio

PseudoCódigo

```
Si PrecioIncluyeIVA Y Calculo Precio/Beneficio<No Relacionado entonces
   TablaConfiguracionIVA.CambioEmpresa (COMPANYNAME)
   TablaConfiguracionIVA.Coge (IVAPrecioProd, IVAProducto)
   Caso TablaConfiguracionIVA.TipoCalculoIVA
         ReservaIVA: TablaConfiguracionIVA.VAT+EC=0
         ImpuestoVenta: Error ('No se puede Calcular IVA')
Sino
  Limpiar TablaConfiguracionIVA
Caso Calculo Precio/Beneficio
    Precio=Coste+Beneficio: ConfigExistencias .CambioEmpresa (COMPANYNAME)
                            Posicionar Primer registro ConfigExistencias
                            Si AjusteBf entonces
                               %Bf=ConfigExistencias.PorcentajeBeneficio
                               %BfPTG=ConfigExistencias.PorcentajeBeneficioPTG
                            PrecioVtaNF= (CosteProducto*(%Bf/100)+
                           CosteProducto)*(1+TablaConfiguracioIVA.VAT+EC/100)
PrecioIncluyeIVA=Falso
Si PrecioVtaNF>0 entonces
   PrecioPsico=CalcularPrecioSicologico (PrecioVtaNF)
   PrecioSinRedóndeoPTG= (CosteProducto*(%BfPTG/100)+ CosteProducto)*
```

```
    (1+TablaConfiguracionIVA.VAT+EC/100)
    PrecioPsicoPTG=CalcularPrecioSicologico (PrecioSinRedóndeoPTG)
    PrecioPsicoPTG=PrecioPsicoPortugal/ (1+IVAPTG/100)*
    (1+ (TablaConfiguracionIVA.VAT+EC/100))
    ModificacionAfectaTPV ()
Sino
   PrecioPsico=0
   PrecioPsicoPTG=0
```

Código Original

```
"Price Includes VAT" := TRUE;
IF "Price Includes VAT" AND
   ("Price/Profit Calculation" < "Price/Profit Calculation"::"No Relationship")
THEN
BEGIN
  VATPostingSetup.CHANGECOMPANY('Poly Servicios Logisticos S.L.');
  IF NOT VATPostingSetup.GET("VAT Bus. Posting Gr. (Price)","VAT Prod. Posting Group")
THEN
    VATPostingSetup.INIT;
  CASE VATPostingSetup."VAT Calculation Type" OF
    VATPostingSetup."VAT Calculation Type"::"Reverse Charge VAT":
      VATPostingSetup."VAT+EC %" := 0;
    VATPostingSetup."VAT Calculation Type"::"Sales Tax":
      ERROR(
        'No se pueden calcular precios IVA incluido cuando %1 es %2.',
        VATPostingSetup.FIELDNAME("VAT Calculation Type"),
        VATPostingSetup."VAT Calculation Type");
  END;
END ELSE
  CLEAR(VATPostingSetup);

CASE "Price/Profit Calculation" OF
  "Price/Profit Calculation"::"Price=Cost+Profit":
  BEGIN
    IF "Ajuste Bfº" THEN
    BEGIN
     "Profit %" = InvtSetup."Porcentaje Beneficio"
     "% Bfº Portugal":=InvtSetup."Porcentaje Beneficio Portugal";
    END;
    PrecioVtaNF :=ROUND(("Last Direct Cost" * ("Profit %" / 100)+"Last Direct Cost") * (1 +
VATPostingSetup."VAT+EC %" / 100),0.00001);
  END;
END;
"Price Includes VAT" := FALSE;
IF PrecioVtaNF > 0 THEN
  BEGIN
   PrecioPsico := CalcularPrecioSicologico(ROUND(PrecioVtaNF,0.01,'='));
   PrecioSinRedóndeoPTG:=ROUND(
                    ("Last Direct Cost" * ("% Bfº Portugal" / 100)+"Last Direct Cost") *
                    (1 + (DameIVAPortugal("VAT Prod. Posting Group") / 100))
                    ,0.01);
```

```
  "PrecioPsico
Portugal":=CalcularPrecioSicologico(ROUND(PrecioSinRedóndeoPTG,0.01,'='));
"PrecioPsico Portugal":= "PrecioPsico Portugal" /
                    (1 + (DameIVAPortugal("VAT Prod. Posting Group") / 100)) *
                    (1 + (VATPostingSetup."VAT+EC %" / 100));

  "PrecioPsico Portugal":= ROUND ("PrecioPsico Portugal",0.01);
  ModificacionAfectaTPV ();
 END
ELSE
BEGIN
 PrecioPsico:= 0;
 "PrecioPsico Portugal":=0;
END;
```

Metodo Coste

CódigoOriginal

ActlzrPCoste

No Proveedor

PseudoCódigo

```
Si AntiguoRegistro.NoProveedor<>NoProveedor Y NoProveedor<>'' entonces
    Si TablaProveedor.Coge (NoProveedor) entonces
        TiempoEspera=TablaProveedor.TiempoEspera
```

Código Original

```
IF (xRec."Vendor No." <> "Vendor No.") AND
  ("Vendor No." <> '')
THEN
 IF Vend.GET("Vendor No.") THEN
   "Lead Time Calculation" := Vend."Lead Time Calculation";
```

CodigoFabricante

PseudoCódigo

```
ConfigExistencias.Coger;
Si NO TablaDimension.Coge (ConfigExistencias.DimFabricante, CodigoFabricante)
     TablaFabricante.Coge (CodigoFabricante);
     TablaDimension se inicializa
     TablaDimension.CodigoDimension= ConfigExistencias.DimFabricante;
     TablaDimension.Codigo=CodigoFabricante
     TablaDimension.Nombre=TablaFabricante.Nombre
     Insertar Registro TablaDimension
TablaEmpresa se inicializa
Se posiciona en el primer registro TablaEmpresa
Repetir
    CargaEmpresa (TablaEmpresa.Nombre)
    TablaDefDimension.CambiarEmpresa (TablaEmpresa.Nombre)
    Si No TablaDefDimension.Coger (27, No, ConfigExistencias.DimFabricante)
```

```
        TablaDefDimension se inicializa
        TablaDefDimension.IDTabla=27
        TablaDefDimension.No=No
        TablaDefDimension.CodigoDimension= ConfigExistencias.DimFabricante;
        TablaDefDimension.CodigoValDimension=CodigoFabricante
        TablaDefDimension.ValorRegistro=MismoCodigo
         Insertar Registro TablaDefDimension
    Sino
         TablaDefDimension.CodigoValDimension=CodigoFabricante
         Modificar registro
Hasta registro TablaEmpresa sea vacío
```

Código Original

```
InvtSetup.GET;
InvtSetup.TESTFIELD(InvtSetup."Cód. Dimensión Fabricante");
IF NOT DimVal.GET(InvtSetup."Cód. Dimensión Fabricante","Manufacturer Code") THEN
BEGIN
  Manuf.GET("Manufacturer Code");
  DimVal.INIT;
  DimVal.VALIDATE("Dimension Code",InvtSetup."Cód. Dimensión Fabricante");
  DimVal.VALIDATE(Code,"Manufacturer Code");
  DimVal.VALIDATE(Name,Manuf.Name);
  DimVal.INSERT(TRUE);
  END;

rcdEmpresa.RESET;
rcdEmpresa.FIND('-');
REPEAT
 cduControlGlobal.CargaEmpresa(rcdEmpresa.Name);
 DefDim.CHANGECOMPANY(rcdEmpresa.Name);
 IF NOT DefDim.GET(27,"No.",InvtSetup."Cód. Dimensión Fabricante") THEN BEGIN
    DefDim.INIT;
    DefDim.VALIDATE("Table ID",27);
    DefDim.VALIDATE("No.","No.");
    DefDim.VALIDATE("Dimension Code",InvtSetup."Cód. Dimensión Fabricante");
    DefDim.VALIDATE("Dimension Value Code","Manufacturer Code");
    DefDim.VALIDATE("Value Posting",DefDim."Value Posting"::"Same Code");
    DefDim.INSERT(TRUE);

 END ELSE
 BEGIN
  DefDim.VALIDATE("Dimension Value Code","Manufacturer Code");
  DefDim.MODIFY;
 END;
UNTIL rcdEmpresa.NEXT=0;
```

Funciones

CalcularPrecioSicologico (PrecioVtaNF)

PseudoCódigo

```
PDecimal=Formateo (PrecioVtaNF,0,'<decimals,3>')
PEntera=Formateo (PrecioVtaNF,0,'<integer>)
PDecimal2=CopiaCadena (PDecimal,2,3)
```

```
Evaluar (PEnt, PEntera)
Evaluar (PDec, PDecimal2)

Si PEnt<3 entonces
   PrecioPsico=Redóndear (PrecioVtaNF, 0.05,’>’)
Sino Si PEnt<30 entonces
   Si PDec<=10 Y PDec>=0 entonces
      PrecioPsico=PEnt;
  Si PDec<=35 Y PDec>=11 entonces
      PrecioPsico=PEnt+0.25;
  Si PDec<=65 Y PDec>=36 entonces
      PrecioPsico=PEnt+0.50;
  Si PDec<=85 Y PDec>=66 entonces
      PrecioPsico=PEnt+0.75;
 Si PDec<=99 Y PDec>=86 entonces
      PrecioPsico=PEnt+0.99;
Sino Si PEnt>=30 entonces
  Si PDec<=75 entonces
      PrecioPsico=PEnt-1+0.99
  Sino Si CopiaCadena (PEnt, LongitudCadena)=’9’ entonces
      PrecioPsico=PEnt+0.99
  Sino
      PrecioPsico=PEnt+1+0.99
```

Código Original

```
// -001
PDectxt := FORMAT(Precio,0,'<decimals,3>');
PEntxt :=  FORMAT(Precio,0,'<integer>');
PDectxt2 := COPYSTR(PDectxt,2,3);
IF EVALUATE(PEnt,PEntxt) THEN
IF EVALUATE(PDec,PDectxt2) THEN
IF PEnt < 3 THEN
BEGIN
     PrecioPsico := ROUND(Precio,0.05,'>');
END
ELSE IF PEnt < 30 THEN
BEGIN
  IF (PDec <= 10) AND (PDec >=0) THEN
    PrecioPsico := PEnt;

  IF (PDec <= 35) AND (PDec >=11) THEN
    PrecioPsico := PEnt+0.25;

  IF (PDec <= 65) AND (PDec >=36) THEN
    PrecioPsico := PEnt+0.50;

  IF (PDec <= 85) AND (PDec >=66) THEN
     PrecioPsico := PEnt+0.75;

  IF (PDec <= 99) AND (PDec >=86) THEN
     PrecioPsico := PEnt+0.99;
END
ELSE  IF PEnt >= 30 THEN
BEGIN
```

```
  IF (PDec <= 75) THEN
    PrecioPsico := PEnt-1 + 0.99
  ELSE IF COPYSTR(FORMAT(PEnt),STRLEN(FORMAT(PEnt))) = '9' THEN
      PrecioPsico := PEnt + 0.99
  ELSE
      PrecioPsico := PEnt+1 + 0.99;
END;
```

ModificaciónAfectaTPV ()

PseudoCódigo

```
Si (Descripción<>AntiguoRegistro.Descripcion) O
   (Descripcion2<>AntiguoRegistro.Descripcion2) O
   (CodigoProveedor<>AntiguoRegistro.CodigoProveedor) O
   (IVAProducto<>AntiguoRegistro.IVAProducto) O
   (PrecioPsico<>AntiguoRegistro.PrecioPsico) O
   (PrecioPsicoPTG<>AntiguoRegistro.PrecioPsicoPTG) entonces
      FechaUltimaMod=HOY
Si (PrecioPsico<>AntiguoRegistro.PrecioPsico) O
    (PrecioPsicoPTG<>AntiguoRegistro.PrecioPsicoPTG) entonces
        FechaUltimaModPrecio=HOY
        Si NO OFERTAPVP Y NO OFERTAPVPPORTUGA entonces
            Si FechaCreacion<>TODAY entonces
                FechaVariacionPrecio=HOY
```

Código Original

```
IF (Description <> xRec.Description) OR
  ("Description 2" <> xRec."Description 2") OR
  ("Vendor Item No." <> xRec."Vendor Item No.") OR
  ("VAT Prod. Posting Group" <> xRec."VAT Prod. Posting Group") OR
  (PrecioPsico <> xRec.PrecioPsico) OR
  ("PrecioPsico Portugal" <> xRec."PrecioPsico Portugal") THEN
BEGIN
  "Fecha Ultima Modif TPV":=CURRENTDATETIME();
END;
IF (PrecioPsico <> xRec.PrecioPsico) OR ("PrecioPsico Portugal" <> xRec."PrecioPsico
Portugal") THEN
BEGIN
  "Fecha Ultima Modif Precio":=CURRENTDATETIME();
  IF      (NOT      DamePVPOferta2('PVP',"No.",TODAY,nvPrecio))      AND      (NOT
DamePVPOferta2('PVPPORTUGA',"No.",TODAY,nvPrecio)) THEN
  BEGIN
    IF "Fecha Creación" <> TODAY THEN
    BEGIN
     "Fecha Variacion Precio Final":=CURRENTDATETIME();
```

ActlzrPCoste

PseudoCódigo

```
Si MetodoCoste=Estandar entonces
      CosteUnitario=CosteEstandar
      %CosteIndirecto=0
```

Sino

CalcularCoste (Rec, nvCoste, CostePromedio)

Si nvCoste<>0 entonces

CosteUnitario=nvCoste

Código Original

```
IF "Costing Method" = "Costing Method"::Standard THEN BEGIN
  "Unit Cost" := "Standard Cost";
  "Indirect Cost %" := 0;
END ELSE
BEGIN
   ItemCostMgt.CalculateAverageCost(Rec,nvCostePromedio,AverageCostACY);
   IF nvCostePromedio <> 0 THEN
     "Unit Cost" := nvCostePromedio;
END;
```

CalcularCoste (Rec, nvCoste, CostePromedio)

PseudoCódigo

```
TablaMovimientoValor se inicializa
TablaMovimientoValor se filtra NoProducto sea Rec.NoProducto
TablaMovimientoValor se filtra FechaValoracion sea Rec.FiltroFecha
TablaMovimientoValor se filtra CodAlmacen sea Rec.FiltroAlmacen
TablaMovimientoValor se filtra CodigoVariante sea Rec.CodigoVariante
TablaMovimientoValor se CalculaSuma (Cantidad Movimiento Producto,
ImporteCosteActual, ImporteTotalCosteActual, ImporteCosteEsperado,
ImporteTotalCosteEsperado)
CantidadPromedio=CantidadMovimientoProducto
nvCoste=ImporteCosteActual+ImporteCosteEsperado
CostePromedio=ImporteTotalCosteActual+ImporteTotalCosteEsperado
Si CantidadPromedio<>0 entonces
    nvCoste=nvCoste/CantidadPromedio
    CostePromedio=CostePromedio/CantidadPromedio
Si CantidadPromedio<0 entonces
    nvCoste=0
ConfiguracionEmpresa.Coge;
Si ConfiguracionEmpresa.MonedaAdicional=’’ O CostePromedio<0 entonces
    CostePromedio=0
Sino
    nvCoste=0
    CostePromedio=0
Si CantidadPromedio<=0 entonces
    EXIT (FALSE)
EXIT (TRUE);
```

Código Original

```
WITH ValueEntry DO BEGIN
  RESET;
  SETCURRENTKEY("Item No.","Valuation Date","Location Code","Variant Code");
  SETRANGE("Item No.",Item."No.");
  SETFILTER("Valuation Date",Item.GETFILTER("Date Filter"));
```

```
  SETFILTER("Location Code",Item.GETFILTER("Location Filter"));
  SETFILTER("Variant Code",Item.GETFILTER("Variant Filter"));
  CALCSUMS(
    "Item Ledger Entry Quantity",
    "Cost Amount (Actual)",
    "Cost Amount (Actual) (ACY)",
    "Cost Amount (Expected)",
    "Cost Amount (Expected) (ACY)");
  AverageQty := "Item Ledger Entry Quantity";
  AverageCost := "Cost Amount (Actual)" + "Cost Amount (Expected)";
  AverageCostACY := "Cost Amount (Actual) (ACY)" + "Cost Amount (Expected) (ACY)";
END;
IF AverageQty <> 0 THEN BEGIN
  AverageCost := AverageCost / AverageQty;
  AverageCostACY := AverageCostACY / AverageQty;

  IF AverageCost < 0 THEN
    AverageCost := 0;
  GLSetup.GET;
  IF (GLSetup."Additional Reporting Currency" = '') OR (AverageCostACY < 0) THEN
    AverageCostACY := 0;
END ELSE BEGIN
  AverageCost := 0;
  AverageCostACY := 0;
END;
IF AverageQty <= 0 THEN
  EXIT(FALSE);
EXIT(TRUE);
```

4.2.3 Ventas

En este apartado se explica cómo se genera un pedido mediante una hoja de Excel y cómo se calcula la tarifa de los productos dependiendo del tipo cliente, indicado en el pedido

Además, se describe cómo se generan facturas entre empresas, debido a que dentro de la empresa principal existen otras empresas a las cuales se sirve mercancía y, por lo tanto, se debe crear una factura en la empresa que lo ha servido. Por otro lado, en la empresa que recibe la mercancía se crea una factura de compra, para que quede constancia de la recepción de la mercancía. Por último, se explica cómo se crean tarifas especiales para productos.

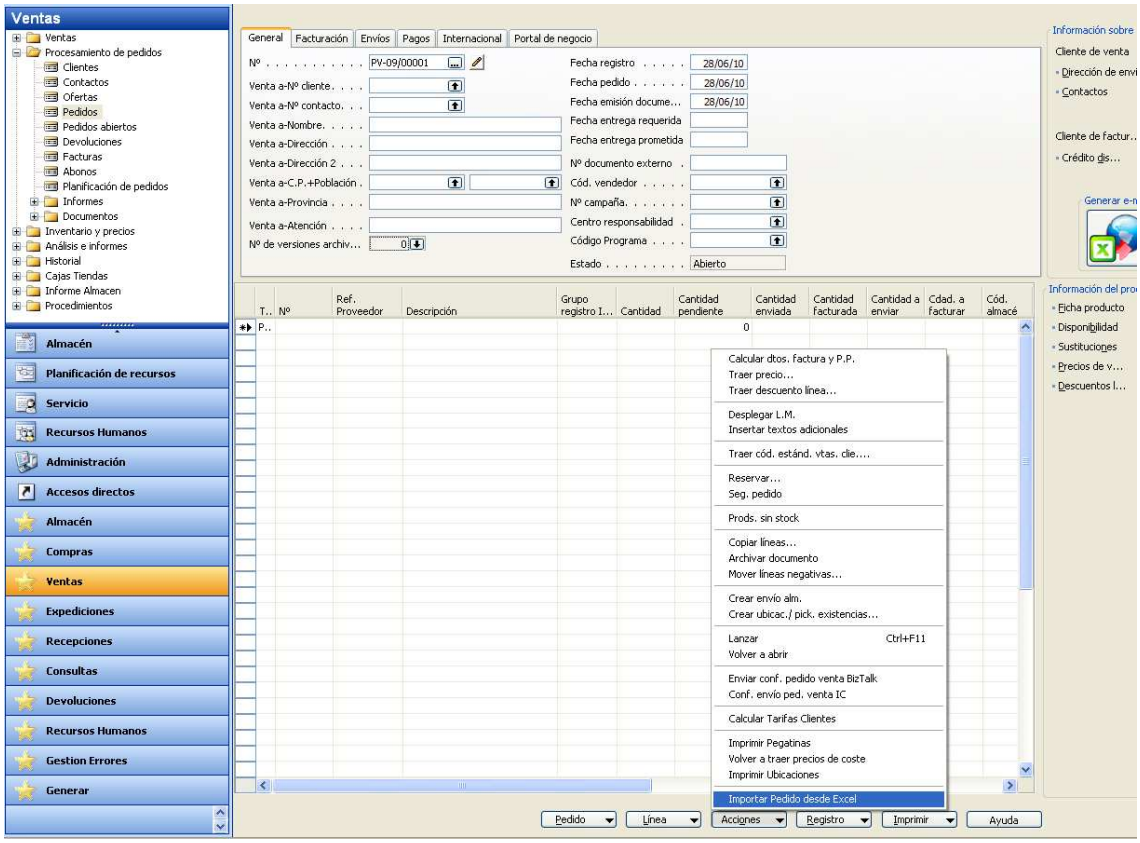


Ilustración 35 Crear Pedido Venta a partir de Hoja de Excel

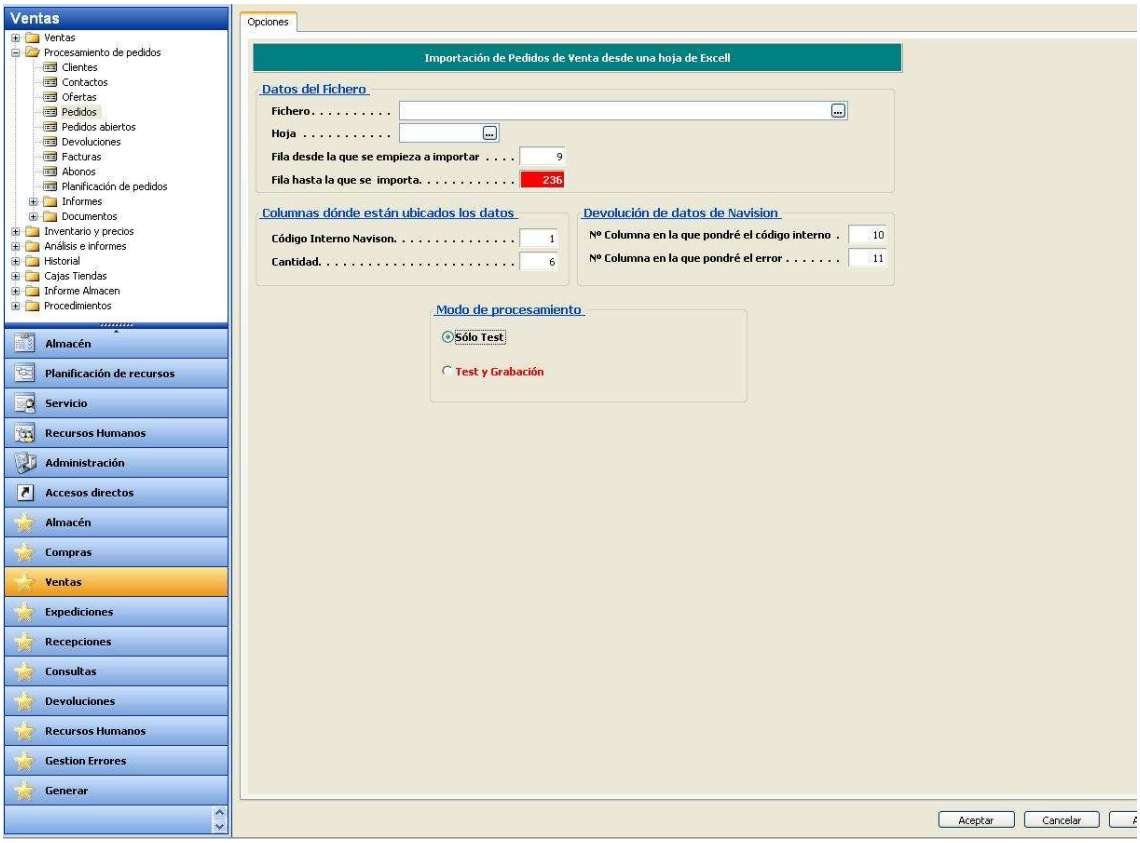


Ilustración 36 Indicar una serie de datos acerca de la hoja de Excel

Importar Pedido Excel (optPedido)

PseudoCódigo

```
Si optPedido='' entonces
    ERROR ('Llamada Incorrecta');
Si FilaDesde>FilaHasta entonces
   ERROR ('Datos incorrectos');
Si no crea aplicación Excel entonces
   ERROR ('Error al acceder a excel');
AplicaciónExcel.Visible=Verdadero
LibroExcel=AplicacionExcel.AbrirLibro (optFichero)
HojaExcel=LibroExcel.CogerHoja (NombreHoja)
HojaExcel se activa
nvLin:=0
LineasVentas se inicializa
LineasVentasAux se filtra TipoDocumento sea Pedido
LineasVentasAux se filtra NoDocumento sea optPedido
Si encuentra ultimo registro LineasVentasAux entonces
   nvLin=LineasVentasAux.NoLinea

For i=FilaDesde hasta Fila Hasta
     //Capturar Datos del Excel
     CodInt=Excell_Dame (i, optCodInt)
     Cantidad=Excell_Dame (i, optColCant)
     cvError=''
     QuitaEspacio (CodInt)
     bError=Falso
     NumCant=0;
     Si longitud(CodInt)=4 entonces
        CodInt='0'+CodInt
    Si Cantidad=0 O Cantidad='' entonces
        bError=Verdadero
        cvError='Cantidad 0'
    Si CodInt<>'' Y No TablaProducto.Coge (CodInt) entonces
       bError=Verdadero
       cvError='Producto no encontrado');
   Si NO bError entonces
       Evaluar (NumCant, Cantidad)
       nvLin+=1
      LineasVenta se inicializa
      LineasVenta.TipoDocumento=Pedido
      LineasVenta.NoDocumento=optPedido
      LineasVenta.Tipo=Producto
      LineasVenta.No=CodInt
      LineasVenta.Cantidad=NumCant
      Insertar registro LineasVenta
      cvError='OK'
  Si optColCodDev<>0 entonces
       Excell_Pinta(i,optColCodDev,CodInt,Falso,Falso)
  Si optColError<>0 entonces
     Excell_Pinta (i, optColError, cvError, Falso, Falso)
Limpiar(AplicacionExcel)
```

Código Original

```
IF optPedido = '' THEN
  ERROR('Error. Este informe sólo puede ser llamado desde un pedido de venta - Acciones - Importar Pedido');

IF optFilaDese > optFilaHasta THEN
  ERROR('Revisa las filas desde y hasta');


IF NOT CREATE(ApllExcel,TRUE) THEN
   ERROR('ERROR AL ACCEDER A EXCELL.\'+
        'Se ha producido un error al instanciar Excell');

ApllExcel.Visible(TRUE);
LibroExcel := ApllExcel.Workbooks._Open(optFichero);
HojaExcel := LibroExcel.Worksheets.Item(SheetName);
HojaExcel.Activate;

nvLin:=0;
rcdLinVentaAux.RESET;
rcdLinVentaAux.SETRANGE("Document Type",rcdLinVentaAux."Document Type"::Order);
rcdLinVentaAux.SETRANGE(rcdLinVentaAux."Document No.",optPedido);

IF rcdLinVentaAux.FINDLAST THEN
  nvLin:=rcdLinVentaAux."Line No.";

FOR i:=optFilaDese TO optFilaHasta DO
BEGIN
 // Capturamos los datos de excell.

 ExcelCodInt:=Excell_Dame(i,optColCodInt);
 ExcelCdad:=Excell_Dame(i,optColCant);
 cvError:='';

 QuitaEspacio(ExcelCodInt);
 bError:=FALSE;
 ExcelNumCdad:=0;

 IF STRLEN(ExcelCodInt) = 4 THEN
    ExcelCodInt:='0'+ExcelCodInt;

 IF (ExcelCdad='') OR (ExcelCdad='0') THEN
 BEGIN
   bError:=TRUE;
   cvError:='Cantidad 0';
 END;

 IF (ExcelCodInt <> '') AND (NOT rcdProducto.GET(ExcelCodInt)) THEN
 BEGIN
  bError:=TRUE;
  cvError:='Producto no encontrado';
 END;

 IF NOT bError THEN
```

```
 BEGIN
   EVALUATE(ExcelNumCdad,ExcelCdad);
   nvLin+=1;
   rcdLinVenta.INIT;
   rcdLinVenta."Document Type":=rcdLinVenta."Document Type"::Order;
   rcdLinVenta."Document No.":=optPedido;
   rcdLinVenta."Line No.":=nvLin;
   rcdLinVenta.Type:=rcdLinVenta.Type::Item;
   rcdLinVenta.VALIDATE("No.",ExcelCodInt);
   rcdLinVenta.VALIDATE(Quantity,ExcelNumCdad);
   rcdLinVenta.INSERT(TRUE);
   cvError:='OK';
 END;
 IF optColCodDev <> 0 THEN
    Excell_Pinta(i,optColCodDev,ExcelCodInt,FALSE,FALSE);

 IF optColError <> 0 THEN
    Excell_Pinta(i,optColError,cvError,FALSE,FALSE);
END;
CLEAR(ApllExcel);
```

Excell_Dame (Fila, Columna)

PseudoCódigo

```
NumFila=Fila
Excell_Filas ( )
NumColumna=Columna
Excell_Columnas ()
Devuelve (HojaExcel.rango(xColID+XRowID).Value)
```

Código Original

```
NumFila := nFila;
Excell_Filas;
NumColumna :=nColumna;
Excell_Columnas;
EXIT(FORMAT(HojaExcel.Range(xlColID + xlRowID).Value));
```

Excell_Filas ()

PseudoCódigo

```
xlRowID=’’
Si NumFila<>0 entonces
   xlRowID=NumFila
```

Código Original

```
xlRowID := '';
IF NumFila <> 0 THEN
  xlRowID := FORMAT(NumFila);
```

Excell_Columnas ()

PseudoCódigo

```
xlColID=’’
Si NunColumna<>0 entonces
    x=NumColumna-1
   c=65+x MOD 26
   xlColID[10]=c
   i=10
  Mientras x>25 entonces
       x := x DIV 26;
       i := i - 1;
      c := 64 + x MOD 26;
      xlColID[i] := c;
For x=i hasta 10
      xlColID[1+x-i] := xlColID[x];
```

Código Original

```
xlColID := '';
IF NumColumna <> 0 THEN BEGIN
  x := NumColumna - 1;
  c := 65 + x MOD 26;
  xlColID[10] := c;
  i := 10;
  WHILE x > 25 DO BEGIN
    x := x DIV 26;
    i := i - 1;
    c := 64 + x MOD 26;
    xlColID[i] := c;
  END;
  FOR x := i TO 10 DO
    xlColID[1+x-i] := xlColID[x];
END;
```

Excell_Pinta (Fila, Columna, Texto, Negrita, Cursiva)

PseudoCódigo

```
NumeroFila=Fila
Excell_Filas
NumeroColumna=Columna
Excell_Columnas
HojaExcel.Rango (xColID+xRowID).Valor=Texto
HojaExcel.Rango (xColID+xRowID).Fuente.Negrita=Negrita
HojaExcel.Rango (xColID+xRowID).Fuente.Cursiva=Cursiva
```

Código Original

```
NumFila := nFila;
Excell_Filas;
NumColumna :=nColumna;
Excell_Columnas;
HojaExcel.Range(xlColID + xlRowID).Value := cValor;
HojaExcel.Range(xlColID + xlRowID).Font.Bold := bNegrita;
HojaExcel.Range(xlColID + xlRowID).Font.Italic := bCursiva;
```

CalcularTarifaCliente(CabVenta,Cliente,LinVenta)

En cada pedido de venta existe la opción de calcular el precio de venta de cada uno de los productos, teniendo en cuenta a que tipo de cliente se está generando el pedido. Entonces, el sistema se recorre cada una de las líneas del pedido y va calculando el precio para cada una. Para el cálculo del precio se hace lo siguiente:

En primer lugar, el sistema mira si para ese producto existe una tarifa específica para el cliente que se configuró en la cabecera del pedido. En caso de existir, el precio de venta corresponde a esa tarifa. En caso de no existir, lo siguiente que mira es una tabla donde se han configurado una serie de descuentos según el proveedor y fabricante correspondientes al producto. Una vez hallado el descuento a realizar, se debe calcular el precio de venta que se aplica al producto. Por otro lado, se calcula el margen que obtendría el cliente vendiendo el producto a ese precio. En caso de que el margen cumpla una serie de requisitos, al precio de venta calculado anteriormente se aplica un descuento adicional.

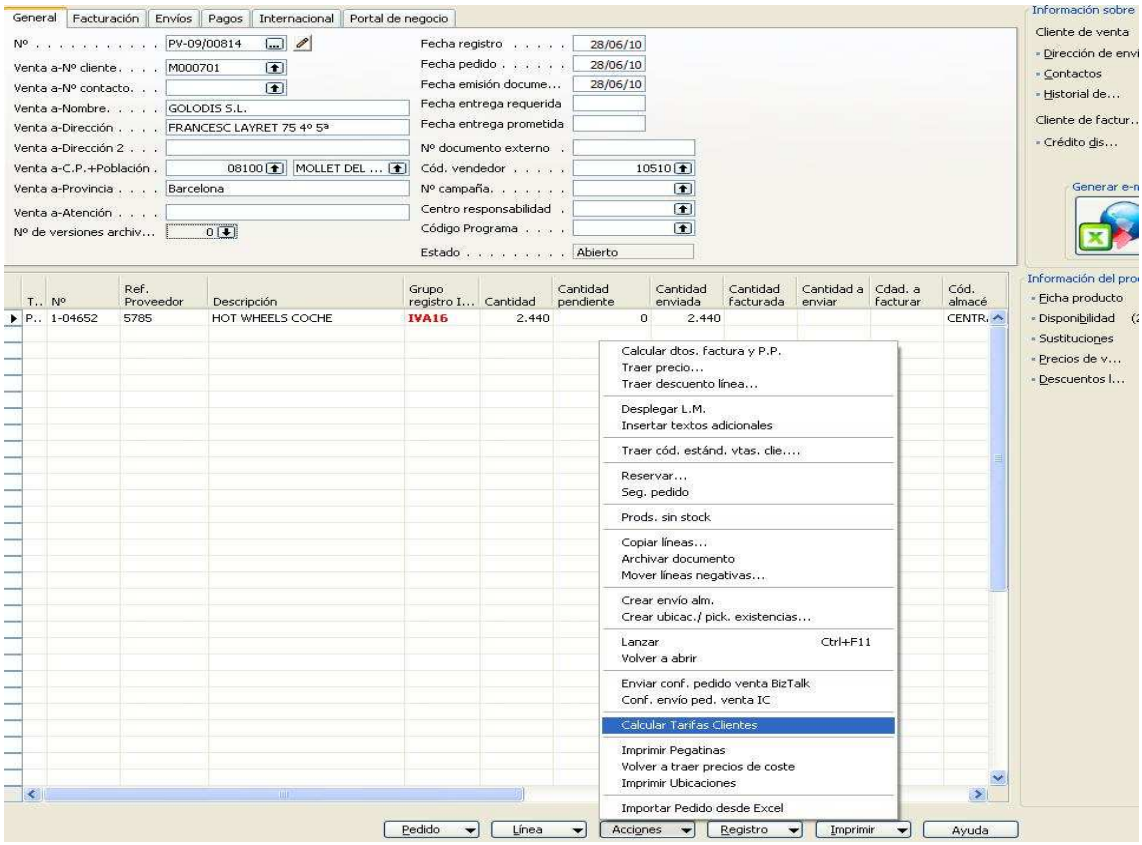


Ilustración 37 Pedido de Venta dónde se muestra la opción de Calcular Tarifa Cliente

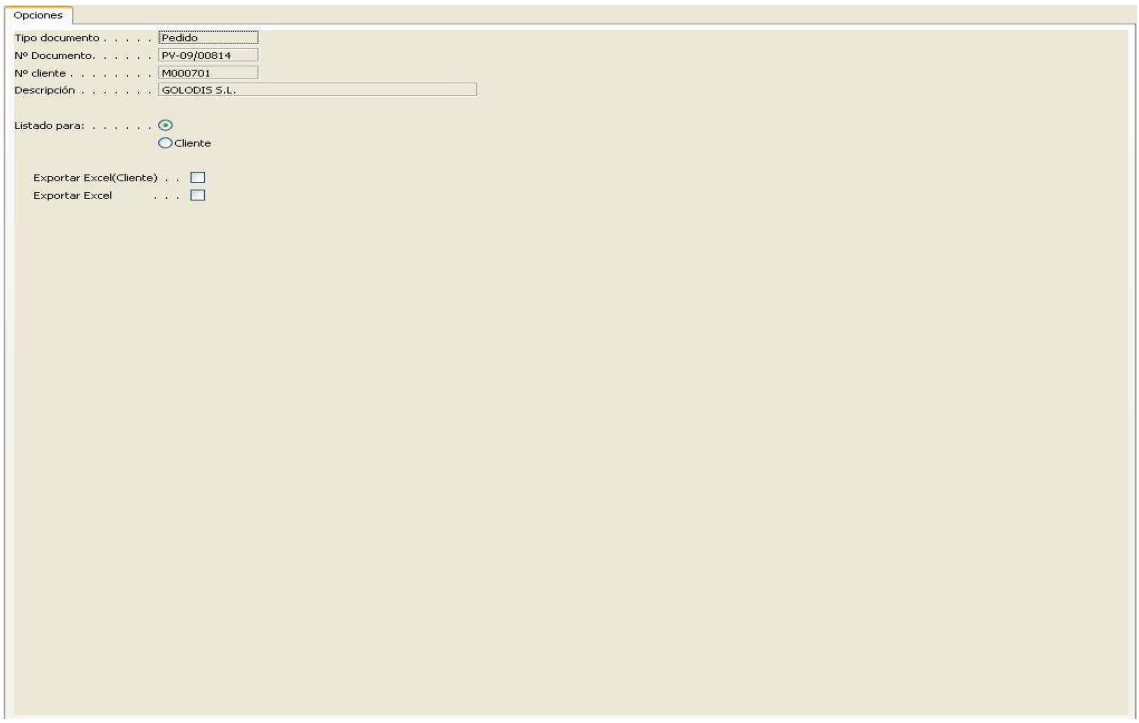


Ilustración 38 Funcion Calculo Tarifa Cliente dónde se debe indicar que se quiere mostrar

PseudoCódigo

CabeceraVenta se filtra TipoDocumento sea Pedido
CabeceraVenta se filtra No sea CabVenta.No
Si Cliente.TipoCliente<>Asociado y Cliente.TipoCliente<>Franquicia entonces
    Error (‘No disponible esta opcion)
TablaProducto.Coge (LinVenta.No)
PrecioFranquicia=LinVenta.PrecioVenta
PrecioCostePoly=LinVenta.CosteUnitario
PrecioPolyconIVA=TablaProducto.PrecioPsico
caso=’No hay condiciones’
Tarifa=Falso
Especial=Falso
TablaIVA se inicializa
TablaIVA se filtra GrupoRegNegocio sea TablaProducto.GrupoRegNegocio
TablaIVA se filtra GrupoRegProd sea TablaProducto.GrupoRegProd
Si no encuentra registros TablaIVA entonces
    Salta registro;
aplicar=1+(TablaIVA.VAT+EC/100);
TablaProducto.DameOferta (TablaProducto.No, TODAY, PrecioPolysinIVA);
Si PrecioPolysinIVA=0 entonces
    PrecioPolysinIVA=TablaProducto.PrecioPsico
PrecioPolysinIVA=Redóndear (PrecioPolyconIVA/aplicar, 0.01)
TablaTarifa se inicializa
TablaTarifa se filtra NoProducto sea LinVenta.No
TablaTarifa se filtra Tipo Venta sea Grupo Precio Cliente
TablaTarifa se filtra CodigoVenta sea CabVenta.GrupoPrecioCliente
TablaTarifa se filtra UnidadMedida sea UD.
TablaTarifa se filtra FechaFin sea igual a 0D O mayor o igual CabVenta.FechaPedido
TablaTarifa se filtra FechaInicio sea mayor 0D Y menor o igual CabVenta.FechaPedido
Si encuentra ultimo registro TablaTarifa entonces

```
    LinVenta.PrecioVenta=TablaProducto.CosteMedio
    Si TablaTarifa.PrecioIncluyeIVA=Verdadero entonces
        LinVenta.Importe=(LinVenta.Cantidad*TablaTarifa.PrecioVenta)/
                                (1+ (LinVenta.%IVA/100)
   Sino
       LinVenta.Importe=LinVenta.Cantidad*TablaTarifa.PrecioVenta
   LinVenta.Validar (DescuentoLinea)
   Modificar registro LinVenta
   Especial=TRUE;
   caso= 'Tarifa Especial Cliente'+FORMATEO(Cliente.TipoCliente)
Si No Especial entonces
    TarifaFranquicias se inicializa
    TarifaFranquicias se filtra NoProveedor sea TablaProducto.NoProveedor
    TarifaFranquicias se filtra NoFabricante sea TablaProducto.NoFabricante
    TarifaFranquicias se filtra NoProducto sea TablaProducto.No
    Si encuentra registro TarifaFranquicias entonces
        Tarifa=Verdadero
        caso='Prov:'+Formateo(TarifaFranquicias.NoProveedor)+'Fabricante:'+
              Formateo(TarifaFranquicias.NoFabricante)+'Producto:'+
              Formateo(TarifaFranquicias.NoProducto)
    Sino
        TarifaFranquicias se filtra NoProducto;
        Si encuentra registros TarifaFranquicias entonces
            Tarifa=Verdadero
            caso='Prov:'+Formateo(TarifaFranquicias.NoProveedor)+'Fabricante:'+
                  Formateo(TarifaFranquicias.NoFabricante)
        Sino
            TarifaFranquicias se filtra NoFabricante;
            TarifaFranquicias se filtra Clase sea TablaProducto.Subfamilia
            Si encuentra registro TarifaFranquicias entonces
                Tarifa=Verdadero
                caso='Prov:'+Formateo(TarifaFranquicias.NoProveedor)+'Familia:'+
                      Formateo(TarifaFranquicias.Clase)
            Sino
                TarifaFranquicia se filtra Clase
                Si encuentra registros TarifaFranquicias entonces
                    Tarifa=Verdadero
                    caso='Prov'+Formateo(TarifaFranquica.NoProveedor)
Si Tarifa entonces
        Si Cliente.TipoCliente=Asociado entonces
            nvCoste=TarifaFranquicias.%Dto Asociado
        Sino
           nvCoste=TarifaFranquicias.%Dto Franquicia
caso=caso+' ('+Formateo(nvCoste)+')'
numero=1-(nvCoste/100)
PrecioFranquicia=Redóndeo (PrecioCostePoly/numero, 0.01)
Si TarifaFranquicia.TipoDescuento=Si Rappel dto=0;sino %sobre neto" entonces
    TablaCompras se inicializa
    TablaCompras se filtra NoProducto sea TablaProducto.No
    TablaCompras se filtra NoProveedor sea TablaProducto.NoProveedor
    Si no encuentra registros TablaCompras O TablaCompras.DtoLinea=0 entonces
         caso=caso+'Rappel'
         PrecioFranquicia=TablaProducto.CosteMedio
    MargenFranquicia= (1-(PrecioFranquicia/PrecioPolysinIVA))*100
```

```
    Si Cliente.TipoCliente=Asociado entonces
        Si MargenFranquicia<0 entonces
            PrecioFranquicia=Redóndeo(PrecioPolysinIVA-(PrecioPolysinIVA*4/100),
                                0.01);
            caso=caso+'Margen<0 (PVP-4%)';
        Si MargenFranquicia>0 Y MargenFranquicia<5 entonces
            PrecioFranquicia=Redóndeo(PrecioFranquicia-(PrecioFranquicia*2/100),0.01)
            caso=caso+'Margen 0-5 (Dto2%)'
       Si MargenFranquicia>5 Y MargenFranquicia<10 entonces
            PrecioFranquicia=Redóndeo(PrecioFranquicia-(PrecioFranquicia*1/100),0.01)
            caso=caso+'Margen 5-10 (Dto1%)'
       MargenFranquicia=(1-(PrecioFranquicia/PrecioPolysinIVA))*100
   Si Cliente.TipoCliente=Franquicia entonces
        Si MargenFranquicia<0 entonces
            PrecioFranquicia=Redóndeo(PrecioPolysinIVA-(PrecioPolysinIVA*5/100),
                                0.01);
            caso=caso+'Margen<0 (PVP-5%)';
        Si MargenFranquicia>0 Y MargenFranquicia<5 entonces
            PrecioFranquicia=Redóndeo(PrecioFranquicia-(PrecioFranquicia*3/100),0.01)
            caso=caso+'Margen 0-5 (Dto2%)'
       Si MargenFranquicia>5 Y MargenFranquicia<10 entonces
            PrecioFranquicia=Redóndeo(PrecioFranquicia-(PrecioFranquicia*2/100),0.01)
            caso=caso+'Margen 5-10 (Dto1%)'
       MargenFranquicia=(1-(PrecioFranquicia/PrecioPolysinIVA))*100
MargenPoly=(1-(PrecioCostePoly/PrecioFranquicia))*100
LinVenta.PrecioVenta=PrecioFranquicia
Modificar Registro LinVenta
```

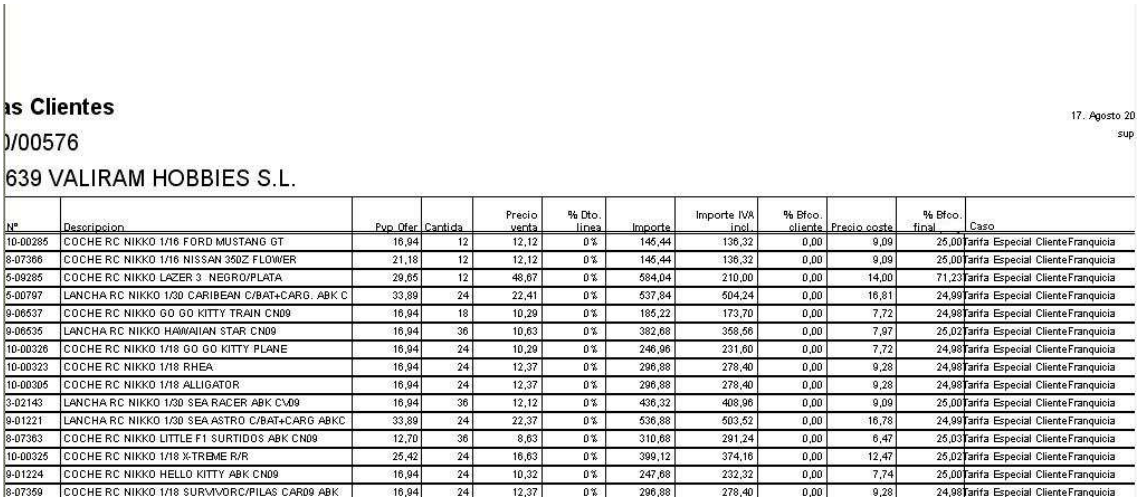
Código Original

```
"Sales Header".SETRANGE("Document Type",CabVenta."Document Type");
"Sales Header".SETFILTER("No.",CabVenta."No.");

IF (rcdCliente."Tipo Cliente"<>rcdCliente."Tipo Cliente"::Asociado) AND
  (rcdCliente."Tipo Cliente"<>rcdCliente."Tipo Cliente"::Franquicia) THEN
  ERROR('No disponible esta opcion');

rcdProducto.GET("Sales Line"."No.");
PrecioFranquicia:="Sales Line"."Unit Price";
PrecioCostePoly:="Sales Line"."Unit Cost (LCY)";
PrecioPVPPolyconIVA:=rcdProducto.PrecioPsico;
caso:='No hay condiciones';
Tarifa:=FALSE;
Especial:=FALSE;
rcdIva.RESET;
rcdIva.SETRANGE(rcdIva."VAT Bus. Posting Group",rcdProducto."VAT Bus. Posting Gr.
(Price)");
rcdIva.SETRANGE(rcdIva."VAT Prod. Posting Group",rcdProducto."VAT Prod. Posting
Group");
IF NOT rcdIva.FINDFIRST THEN
  CurrReport.SKIP;
aplicar:=1+(rcdIva."VAT+EC %"/100);

rcdProducto.DamePVPOferta(rcdProducto."No.",TODAY,PrecioPVPPolyconIVA);
IF PrecioPVPPolyconIVA=0 THEN
```

```
   PrecioPVPPolyconIVA:=rcdProducto.PrecioPsico;

PrecioPVPPolysinIVA:=ROUND(PrecioPVPPolyconIVA /aplicar,0.01);
//////////////////////////////////////////////////////////////
// Buscamos Tarifas 'estándar' para ese cliente
//////////////////////////////////////////////////////////////

rcdTarifa.RESET;
rcdTarifa.SETRANGE(rcdTarifa."Item No.","Sales Line"."No.");
rcdTarifa.SETRANGE("Sales Type",rcdTarifa."Sales Type"::"Customer Price Group");
rcdTarifa.SETRANGE("Sales Code",CabVenta."Customer Price Group");
rcdTarifa.SETRANGE("Unit of Measure Code",'UD.');
rcdTarifa.SETFILTER("Ending Date",'%1|>=%2',0D,"Sales Header"."Order Date");
rcdTarifa.SETFILTER("Starting Date",'>%1&<=%2',0D,"Sales Header"."Order Date");

IF rcdTarifa.FINDLAST THEN
BEGIN
  "Sales Line".VALIDATE("Sales Line"."Unit Price",ROUND(rcdProducto."Last Direct
Cost",0.01));
   IF rcdTarifa."Price Includes VAT" =TRUE THEN
      "Sales Line".VALIDATE(Amount,("Sales Line"."Quantity (Base)"*rcdTarifa."Unit
Price")/(1+("Sales Line"."VAT %"/100)))
   ELSE
      "Sales Line".VALIDATE(Amount,"Sales Line"."Quantity (Base)"*rcdTarifa."Unit Price");

   "Sales Line".VALIDATE("Line Discount %",ROUND("Sales Line"."Line Discount
%",0.01));
   "Sales Line".MODIFY;
   Especial:=TRUE;
   caso:='Tarifa Especial Cliente'+FORMAT(rcdCliente."Tipo Cliente");
END;

IF NOT Especial THEN
BEGIN
  //////////////////////////////////////////////////////////////
  // Buscamos Tarifas para ese Proveedor + fabricante + producto
  //////////////////////////////////////////////////////////////
  rcdTarifaFran.RESET;
  rcdTarifaFran.SETRANGE("Nº Proveedor",rcdProducto."Vendor No.");
  rcdTarifaFran.SETRANGE("Nº Fabricante",rcdProducto."Manufacturer Code");
  rcdTarifaFran.SETRANGE("Nº Producto",rcdProducto."No.");
  IF rcdTarifaFran. FINDFIRST THEN
  BEGIN
    Tarifa:=TRUE;
    caso:='Prov:'+FORMAT(rcdTarifaFran."Nº Proveedor")+'Fabricante:'
          +FORMAT(rcdTarifaFran."Nº Fabricante")+'Producto:'+FORMAT(rcdTarifaFran."Nº
Producto");
  END
  ELSE
  BEGIN
    //////////////////////////////////////////////////////////////
    // Buscamos Tarifas para ese Proveedor + fabricante
    //////////////////////////////////////////////////////////////
    rcdTarifaFran.SETRANGE("Nº Producto");
```

```
    IF rcdTarifaFran. FINDFIRST THEN
    BEGIN
      Tarifa:=TRUE;
      caso:='Prov:'+FORMAT(rcdTarifaFran."Nº
Proveedor")+'Fabricante:'+FORMAT(rcdTarifaFran."Nº Fabricante");
    END
    ELSE
    BEGIN
     ////////////////////////////////////////////////////////
     // Buscamos Tarifas para ese Proveedor + Familia
     ////////////////////////////////////////////////////////
     rcdTarifaFran.SETRANGE("Nº Fabricante");
     rcdTarifaFran.SETRANGE(rcdTarifaFran.Clase,rcdProducto."Global Dimension 1 Code");
     IF rcdTarifaFran.FINDFIRST THEN
     BEGIN
        Tarifa:=TRUE;
        caso:='Prov:'+FORMAT(rcdTarifaFran."Nº
Proveedor")+'Familia:'+FORMAT(rcdTarifaFran.Clase);
     END
     ELSE
     BEGIN
       rcdTarifaFran.SETRANGE(Clase);
       IF rcdTarifaFran.FINDFIRST THEN
       BEGIN
         Tarifa:=TRUE;
         caso:='Prov:'+FORMAT(rcdTarifaFran."Nº Proveedor");
       END;
     END;
    END;
  END;
END;

IF Tarifa THEN
BEGIN
  IF rcdCliente."Tipo Cliente"=rcdCliente."Tipo Cliente"::Asociado THEN
     nvCoste:=rcdTarifaFran."%Dto Asociado"
  ELSE
     nvCoste:=rcdTarifaFran."% Dto Franquicia";

  caso:=caso+' ('+FORMAT(nvCoste)+')';

  numero:=1-(nvCoste/100);
  PrecioFranquicia:=ROUND(PrecioCostePoly/numero,0.01);

  IF rcdTarifaFran."Tipo Descuento" = rcdTarifaFran."Tipo Descuento"::"Si Rappel dto=0; sino
% Sobre Neto" THEN
  BEGIN
    rcdCompras.RESET;
    rcdCompras.SETRANGE("Item No.",rcdProducto."No.");
    rcdCompras.SETRANGE("Vendor No.",rcdProducto."Vendor No.");
    IF (NOT rcdCompras.FINDFIRST) OR (rcdCompras."Line Discount %" = 0 )THEN
    BEGIN
       caso:=caso+'Rappel';
       PrecioFranquicia:=rcdProducto."Last Direct Cost";
```

```
    END;
  END;
  MargenFranquicia:=(1-(PrecioFranquicia/PrecioPVPPolysinIVA))*100;
  IF rcdCliente."Tipo Cliente"=rcdCliente."Tipo Cliente"::Asociado THEN
  BEGIN
    IF MargenFranquicia<0 THEN
    BEGIN
      PrecioFranquicia:=ROUND(PrecioPVPPolysinIVA-(PrecioPVPPolysinIVA*4/100),0.01);
      caso:=caso+' Margen<0 (PVP-4%)';
    END;
    IF (MargenFranquicia>0) AND (MargenFranquicia<5) THEN
    BEGIN
      PrecioFranquicia:=ROUND(PrecioFranquicia-(PrecioFranquicia*2/100),0.01);
      caso:=caso+' Margen 0-5 (Dto 2%)';
    END;
    IF (MargenFranquicia>5) AND (MargenFranquicia<10) THEN
    BEGIN
      PrecioFranquicia:=ROUND(PrecioFranquicia-(PrecioFranquicia*1/100),0.01);
      caso:=caso+' Margen 5-10 (Dto 1%)';
    END;
    MargenFranquicia:=(1-(PrecioFranquicia/PrecioPVPPolysinIVA))*100;
  END;
  IF rcdCliente."Tipo Cliente"=rcdCliente."Tipo Cliente"::Franquicia THEN
  BEGIN
    IF MargenFranquicia<0 THEN
    BEGIN
      PrecioFranquicia:=ROUND(PrecioPVPPolysinIVA-(PrecioPVPPolysinIVA*5/100),0.01);
      caso:=caso+' Margen<0 (PVP-5%)';
    END;
    IF (MargenFranquicia>0) AND (MargenFranquicia<5) THEN
    BEGIN
      PrecioFranquicia:=ROUND(PrecioFranquicia-(PrecioFranquicia*3/100),0.01);
      caso:=caso+' Margen 0-5 (Dto 3%)';
    END;
    IF (MargenFranquicia>5) AND (MargenFranquicia<10) THEN
    BEGIN
      PrecioFranquicia:=ROUND(PrecioFranquicia-(PrecioFranquicia*2/100),0.01);
      caso:=caso+' Margen 5-10 (Dto 2%)';
    END;
    MargenFranquicia:=(1-(PrecioFranquicia/PrecioPVPPolysinIVA))*100;
  END;
END;
MargenPoly := (1 - (PrecioCostePoly / PrecioFranquicia)) * 100;
PrecioPVPPolysinIVATot:=PrecioPVPPolysinIVATot + (PrecioPVPPolysinIVA*Quantity);
PrecioPVPPolyconIVATot:=PrecioPVPPolyconIVATot + (PrecioPVPPolyconIVA*Quantity);
"Sales Line".VALIDATE("Unit Price",PrecioFranquicia);
"Sales Line".MODIFY;
```



as Clientes

17. Agosto 20

sup

)/00576

639 VALIRAM HOBBIES S.L.

Nº	Descripcion	Pvp. Ofer.	Cantida	Precio venta	% Dto. linea	Importe	Importe IVA incl.	% Bfco. cliente	Precio coste	% Bfco. final	Caso
10-00285	COCHE RC NIKKO 1/16 FORD MUSTANG GT	16,94	12	12,12	0%	145,44	136,32	0,00	9,09	25,00	Tarifa Especial ClienteFranquicia
8-07366	COCHE RC NIKKO 1/16 NISSAN 350Z FLOWER	21,18	12	12,12	0%	145,44	136,32	0,00	9,09	25,00	Tarifa Especial ClienteFranquicia
5-09285	COCHE RC NIKKO LAZER 3 NEGRO/PLATA	29,65	12	48,67	0%	584,04	210,00	0,00	14,00	71,23	Tarifa Especial ClienteFranquicia
5-00797	LANCHA RC NIKKO 1/30 CARIBEAN C/BAT+CARG. ABK C	33,89	24	22,41	0%	537,84	504,24	0,00	16,81	24,99	Tarifa Especial ClienteFranquicia
9-06537	COCHE RC NIKKO GO GO KITTY TRAIN CN09	16,94	18	10,29	0%	185,22	173,70	0,00	7,72	24,98	Tarifa Especial ClienteFranquicia
9-06535	LANCHA RC NIKKO HAWAIIAN STAR CN09	16,94	36	10,63	0%	382,68	358,56	0,00	7,97	25,02	Tarifa Especial ClienteFranquicia
10-00326	COCHE RC NIKKO 1/18 GO GO KITTY PLANE	16,94	24	10,29	0%	246,96	231,60	0,00	7,72	24,98	Tarifa Especial ClienteFranquicia
10-00323	COCHE RC NIKKO 1/18 RHEA	16,94	24	12,37	0%	296,88	278,40	0,00	9,28	24,98	Tarifa Especial ClienteFranquicia
10-00305	COCHE RC NIKKO 1/18 ALLIGATOR	16,94	24	12,37	0%	296,88	278,40	0,00	9,28	24,98	Tarifa Especial ClienteFranquicia
3-02143	LANCHA RC NIKKO 1/30 SEA RACER ABK C\09	16,94	36	12,12	0%	436,32	408,96	0,00	9,09	25,00	Tarifa Especial ClienteFranquicia
9-01221	LANCHA RC NIKKO 1/30 SEA ASTRO C/BAT+CARG ABKC	33,89	24	22,37	0%	536,88	503,52	0,00	16,78	24,99	Tarifa Especial ClienteFranquicia
8-07363	COCHE RC NIKKO LITTLE F1 SURTIDOS ABK CN09	12,70	36	8,63	0%	310,68	291,24	0,00	6,47	25,03	Tarifa Especial ClienteFranquicia
10-00325	COCHE RC NIKKO 1/18 X-TREME R/R	25,42	24	16,63	0%	399,12	374,16	0,00	12,47	25,02	Tarifa Especial ClienteFranquicia
9-01224	COCHE RC NIKKO HELLO KITTY ABK CN09	16,94	24	10,32	0%	247,68	232,32	0,00	7,74	25,00	Tarifa Especial ClienteFranquicia
8-07359	COCHE RC NIKKO 1/18 SURVIVORC/PILAS CAR09 ABK	16,94	24	12,37	0%	296,88	278,40	0,00	9,28	24,98	Tarifa Especial ClienteFranquicia

Ilustración 39 Informe con los datos que requiere para poder saber el beneficio del pedido

Factura InterEmpresa(MovProd,MovProv)

En este apartado, se habla de cómo se crean facturas entre empresas asociadas. Estas facturas se crean debido a que el almacén central envía mercancía a las tiendas, las cuales facturan en otra empresa distinta. Por lo tanto, se debe hacer constancia de ese movimiento de mercancía a nivel contable. Una vez creada la factura en la empresa desde dónde se sirvió la mercancía, se debe generar un pedido de compra en la empresa que lo ha recibido, este proceso se explica más adelante. Cómo muestra la ilustración 40, se debe indicar en que cuenta se imputa el importe de la factura, tanto para las transferencias como para los pedidos directos de proveedor. Además, se debe indicar el porcentaje que se quiere aplicar al coste de la transferencia o pedido. Una vez configurados los parámetros, se ejecuta el informe para generar la factura.

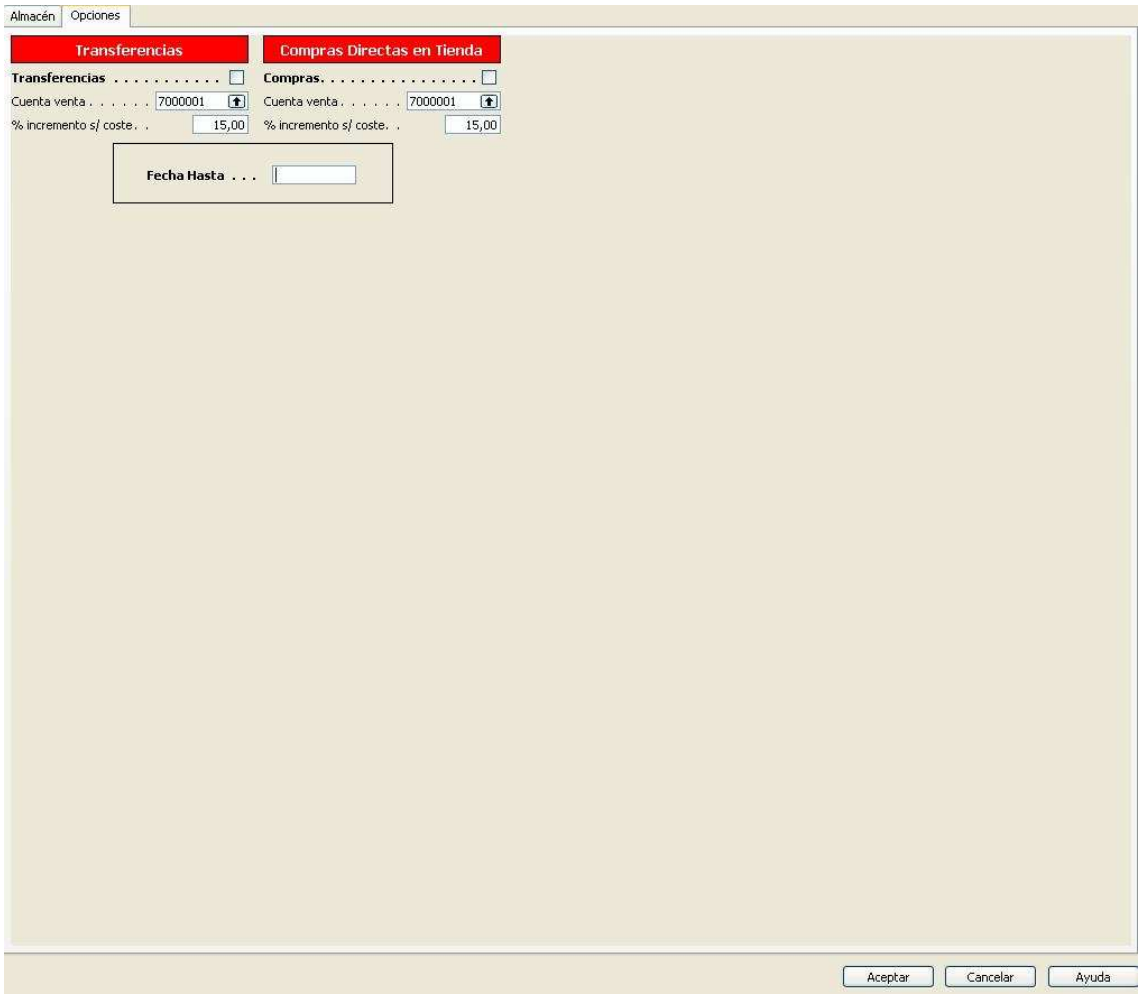


Ilustración 40 Parámetros que se deben configurar para generar la factura interEmpresa

PseudoCódigo

```
//Transferencias
Repetir
  TablaProducto se inicializa
  TablaProducto se filtra No sea MovProd.NoProducto
  Se posiciona en el primer registro TablaProducto
  Si No Cabecera entonces
     CrearCabecera()
     Cabecera=Verdadero
  Si DocumentoAnterior<>MovProd.NoDocumento entonces
     CrearLinea(MovProd)
  Sino
     ModificarLinea(MovProd)
  DocumentoAnterior=MovProd.NoDocumento
   MovProdAux.Coge(MovProd.NoMov)
   MovProdAux.Procesada=Verdadero
   MovProdAux.DocumentoProceso=LinVenta.NoDocumento
   MovProdAux.LineaDocumentoProceso=LinVenta.NoLinea
   Modificar registro MovProd
Hasta registro MovProd sea vacío
//Proveedor
```

```
Repetir
  TablaProducto se inicializa
  TablaProducto se filtra No sea MovProd.NoProducto
  Se posiciona en el primer registro TablaProducto
  Si No Cabecera entonces
    CrearCabecera()
    Cabecera=Verdadero
  Si DocumentoAnterior<>MovProv.NoDocumento entonces
    CrearLinea(MovProv)
  Sino
    ModificarLinea(MovProv)
  DocumentoAnterior=MovProv.NoDocumento
   MovProvAux.Coge(MovProv.NoMov)
   MovProvAux.Procesada=Verdadero
   MovProvAux.DocumentoProceso=LinVenta.NoDocumento
   MovProvAux.LineaDocumentoProceso=LinVenta.NoLinea
   Modificar registro MovProv
Hasta registro MovProd sea vacío
```

Código Original

```
PuntProducto.RESET;
PuntProducto.SETRANGE("No.","Nº producto");
IF PuntProducto.FINDFIRST THEN;

IF NOT Cabecera THEN
BEGIN
  CrearCabecera();
  Cabecera:=TRUE;
END;

IF DocumentoAnterior <> "Item Ledger Entry AUX"."Nº documento" THEN
  CrearLinea("Item Ledger Entry AUX")
ELSE
  ModificarLinea("Item Ledger Entry AUX");
DocumentoAnterior := "Item Ledger Entry AUX"."Nº documento";

//Marcamos el registro cómo procesada
ItemLedAux2.GET("Item Ledger Entry AUX"."Nº mov.");
ItemLedAux2.Procesada := TRUE;
ItemLedAux2."Documento Proceso" := LinVenta."Document No.";
ItemLedAux2."Linea Documento Proceso" := LinVenta."Line No.";
ItemLedAux2.MODIFY;

PuntProducto.RESET;
PuntProducto.SETRANGE("No.","Nº producto");
IF PuntProducto.FINDFIRST THEN;

IF NOT Cabecera THEN
BEGIN
  CrearCabecera();
  Cabecera:=TRUE;
END;

IF DocumentoAnterior <> Proveedor."Nº documento" THEN
```

```
  CrearLinea(Proveedor)
ELSE
  ModificarLinea(Proveedor);
DocumentoAnterior := Proveedor."Nº documento";


//Marcamos el registro cómo procesada
ItemLedAux2.GET(Proveedor."Nº mov.");
ItemLedAux2.Procesada := TRUE;
ItemLedAux2."Documento Proceso" := LinVenta."Document No.";
ItemLedAux2."Linea Documento Proceso" := LinVenta."Line No.";
```

CrearCabecera

PseudoCódigo

```
TablaAlmacen.Coge(Almacen.Codigo)
TablaEmpresa.Coge(TablaAlmacen.NombreEmpresa)
CabeceraVenta se inicializa
CabeceraVenta.No=’’
CabeceraVenta.TipoDocumento=Factura
CabeceraVenta.RegistroNoSeries=’V-FAC2+’
Insertar registro CabeceraVenta
CabeceraVenta.NoCliente=TablaEmpresa.CodCliente
CabeceraVenta.Departamento=TablaAlmacen.CodigoEquivalencia
Modificar registro CabeceraVenta
```

Código Original

```
PuntAlmacen.GET(Location.Code);
PuntEmpresa.GET(PuntAlmacen."Nombre Empresa");

CabVenta.INIT;
CabVenta."No.":=";
CabVenta."Document Type" := CabVenta."Document Type"::Invoice;
CabVenta."Posting No. Series":='V-FAC2+';
CabVenta.INSERT(TRUE);

CabVenta.VALIDATE("Sell-to Customer No.",PuntEmpresa."Cod. cliente");
CabVenta.VALIDATE("Shortcut Dimension 2 Code",PuntAlmacen."Cód. Equivalencia
Antiguo");
CabVenta.MODIFY(TRUE);

COMMIT;
```

CrearLinea(rcdMov)

PseudoCódigo

```
LinVenta se inicializa
LinVenta se filtra TipoDocumento sea Factura
LinVenta se filtra NoDocumento sea CabVenta.No
Si encuentra ultimo registro LinVenta entonces
    numLin=LinVenta.NoLinea
LinVenta se inicializa
LinVenta.TipoDocumento=Factura
```

```
LinVenta.NoDocumento=CabVenta.No
LinVenta.NoLinea=numLin+1000
LinVenta.Tipo=Cuenta
Si rcdMov.TipoMovimiento=Transferencia entonces
    LinVenta.No=CuentaVenta
    LinVenta.PrecioVenta=rcdMov.Cantidad*(TablaProducto.CosteUnitario+
                        TablaProducto.CosteUnitario*IncrementoC/100)))
Sino
    LinVenta.No=CuentaCompras
    LinVenta.PrecioVenta=rcdMov.Cantidad*(TablaProducto.CosteUnitario+
                        TablaProducto.CosteUnitario*IncrementoV/100)))
LinVenta.Descripcion=rcdMov.NoDocumento
LinVenta.Cantidad=1
LinVenta.GrupoRegistroProd=TablaProducto.GrupoRegistroProd
Insertar registro LinVenta
```

Código Original

```
LinVenta.RESET;
LinVenta.SETRANGE("Document Type",LinVenta."Document Type"::Invoice);
LinVenta.SETRANGE("Document No.",CabVenta."No.");
IF LinVenta.FINDLAST THEN
  NumLin := LinVenta."Line No.";

LinVenta.INIT;
LinVenta.RESET;
LinVenta.VALIDATE("Document Type",LinVenta."Document Type"::Invoice);
LinVenta.VALIDATE("Document No.",CabVenta."No.");
LinVenta.VALIDATE("Line No.",NumLin + 1000);
LinVenta.VALIDATE(Type,LinVenta.Type::"G/L Account");
IF rcdMov."Tipo movimiento" = rcdMov."Tipo movimiento"::Transfer THEN
BEGIN
  LinVenta.VALIDATE("No.",CuentaVenta);
  LinVenta.VALIDATE("Unit Price",rcdMov.Cantidad * (PuntProducto."Unit
Cost"+(PuntProducto."Unit Cost"*PorcIncremento/100)));
END
ELSE
BEGIN
  LinVenta.VALIDATE("No.",CuentaCompras);
  LinVenta.VALIDATE("Unit Price",rcdMov.Cantidad * (PuntProducto."Unit
Cost"+(PuntProducto."Unit Cost"*PorcIncrementoC/100)));
END;
LinVenta.VALIDATE(Description,rcdMov."Nº documento");
LinVenta.VALIDATE(Quantity,1);
LinVenta.VALIDATE("VAT Prod. Posting Group",PuntProducto."VAT Prod. Posting Group");
LinVenta.INSERT(TRUE);
COMMIT;
```

ModificarLinea(rcdMov)

PseudoCódigo

```
LinVenta se inicializa
LinVenta se filtra TipoDocumento sea Factura
LinVenta se filtra NoDocumento sea CabVenta.No
LinVenta se filtra Descripción sea rcdMov.NoDocumento
```

```
LinVenta se filtra GrupoRegistroProducto sea TablaProducto.GrupoRegistroProducto
Si encuentra registros LinVenta entonces
    Si rcdMov.TipoMovimiento=Transferencia
        LinVenta.Precio+=rcdMov.Cantidad*(TablaProducto.CosteUnitario+
                        (TablaProducto.CosteUnitario*IncrementoV/100))
   Sino
      LinVenta.Precio+=rcdMov.Cantidad*(TablaProducto.CosteUnitario+
                        (TablaProducto.CosteUnitario*IncrementoC/100))
   Modificar registro LinVenta
   Si LinVenta.PrecioVenta=0
       Borrar registro LinVenta
Sino
  CrearLinea(rcdMov)
```

Código Original

```
LinVenta.RESET;
LinVenta.SETRANGE("Document Type",LinVenta."Document Type"::Invoice);
LinVenta.SETRANGE("Document No.",CabVenta."No.");
LinVenta.SETRANGE(Description,rcdMov."Nº documento");
LinVenta.SETRANGE("VAT Prod. Posting Group",PuntProducto."VAT Prod. Posting
Group");
IF LinVenta.FINDFIRST THEN
  BEGIN
    IF rcdMov."Tipo movimiento" = rcdMov."Tipo movimiento"::Transfer THEN
      LinVenta."Unit Price" += rcdMov.Cantidad * (PuntProducto."Unit
Cost"+(PuntProducto."Unit Cost"*PorcIncremento/100))
    ELSE
      LinVenta."Unit Price" += rcdMov.Cantidad * (PuntProducto."Unit
Cost"+(PuntProducto."Unit Cost"*PorcIncrementoC/100));
    LinVenta.VALIDATE("Unit Price");
    LinVenta.MODIFY;
    IF LinVenta."Unit Price"=0 THEN
       LinVenta.DELETE;
  END
ELSE
  CrearLinea(rcdMov);
```

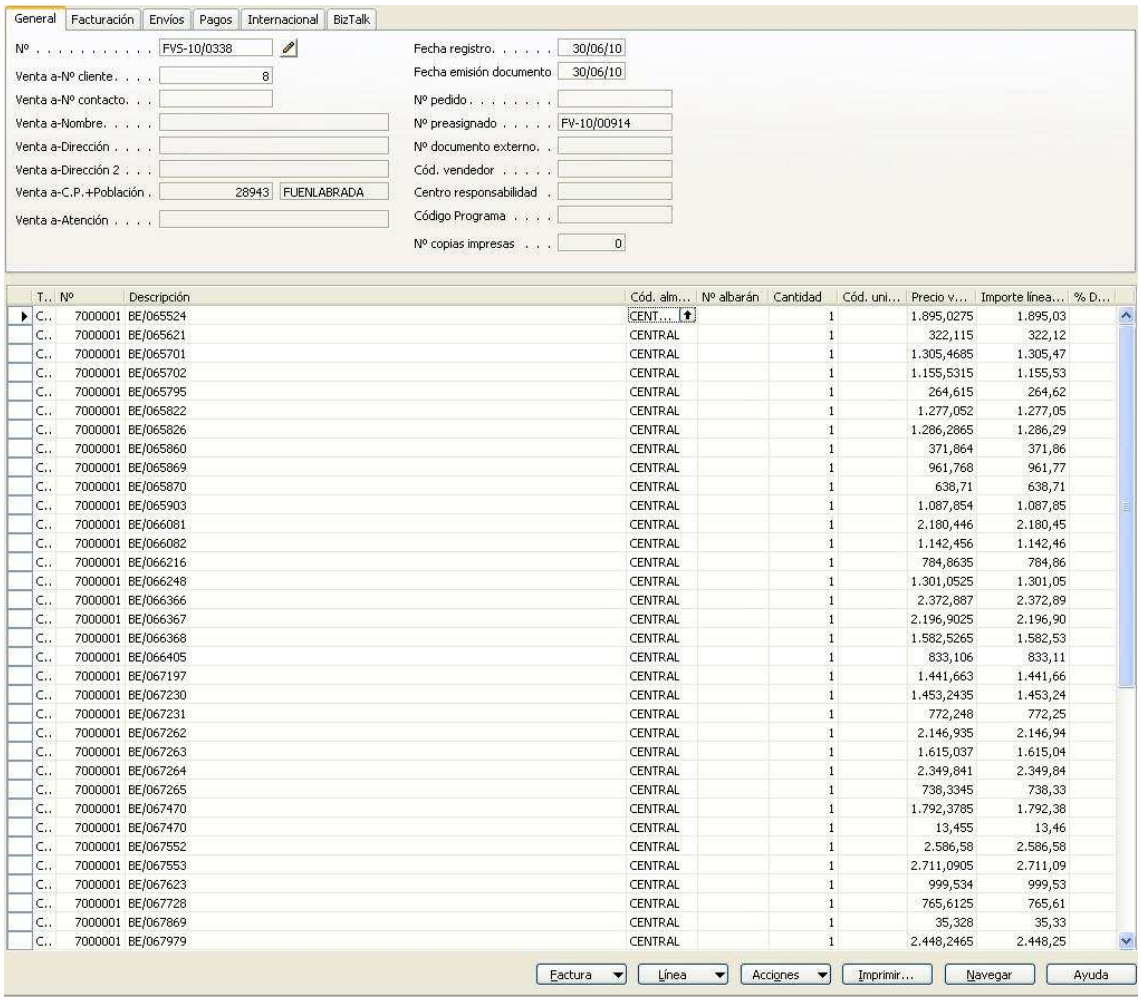


Ilustración 41 Factura InterEmpresa que se crea al ejecutar el informe

Crear Factura Compra(CabVenta)

En este apartado, se explica cómo se genera la factura de compra correspondiente a la factura interempresa creada. En este caso, la factura de compra se crea en la empresa que recibe la mercancía. Entonces, el sistema va acumulando los importes de IVA de cada una de la líneas, comprobando a que tipo de IVA corresponde. Una vez acumulado, se crea la factura con los datos de los distintos IVA.

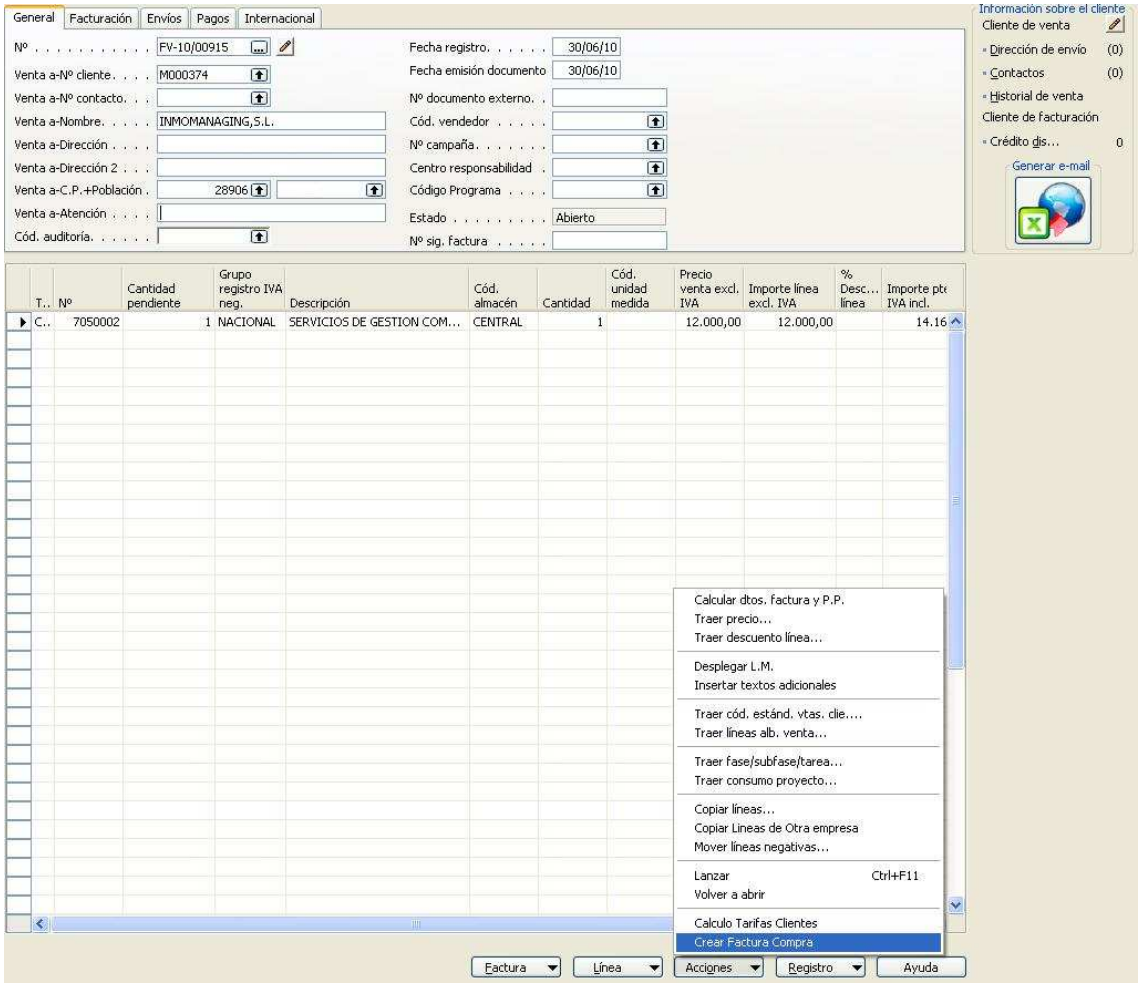


Ilustración 42 Opción dónde se debe clicar para poder generar la factura de compra.

PseudoCódigo

```
TablaEmpresa se inicializa
TablaEmpresa se filtra CodCliente sea CabVenta.NoCliente
Si no encuentra registros TablaEmpresa entonces
      ERROR (‘No existe la empresa’);
CabCompraAux se inicializa
CabCompraAux.CambiaEmpresa(TablaEmpresa.Nombre)
CabCompraAux se filtra TipoDocumento sea Factura
CabCompra se filtra NoFactura sea CabVenta.No
Si encuentra registros CabCompraAux entonces
     ERROR (‘ Ya existe la factura’);
HistCabCompra se inicializa
HistCabCompra.CambiaEmpresa(TablaEmpresa.Nombre)
HistCabCompra se filtra NoFactura sea CabVenta.No
Si encuentra registros HistCabCompra entonces
      ERROR( ‘Ya creo y facturo la correspondiente factura compra’);
LinVenta se inicializa
LinVenta se filtra NoDocumento sea CabVenta.No
Si encuentra registros LinVenta entonces
   Repetir
      Si no TablaIVa.Coge(LinVenta.IdIVA,LinVenta.TipoCalculoIVA,
                          LinVenta.GrupoImpuestos,Falso, LinVenta.Importe>=0)
```

```
            TablaIVA se inicializa
            TablaIVA.IdIVA=LinVenta.IdIVA
            TablaIVA.TipoCalculoIVA=LinVenta.TipoCalculoIVA
            TablaConfigIVA.Coge(LinVenta.GrupoRegIVA, LinVenta.GrupoRegProd)
            TablaIVA.EC=TablaConfig.EC
            TablaIVA.IVA%=TablaConfig.IVA%
            TablaIVA.Modificar=Verdadero
            TablaIVA.Positivo=LinVenta.ImporteLinea>=0
            Insertar registro TablaIVA
        TablaIVA.Cantidad=TablaIVA.Cantidad+LinVenta.Cantidad(Base)
        TablaIVA.ImporteLinea=TablaIVA.ImporteLinea+LinVenta.ImporteLinea
        Si DescuentoFactura entonces
            TablaIVA.DescuentoImporte= TablaIVA.FacturaDescuentoImporte+
                                                    LinVenta.ImporteLinea
        TablaIVA.FacturaDescuentoImporte=TablaIVAFacturaDescuentoImporte+
                                                LinVenta.FacturaDescuentoImporte
        TablaIVA.ImporteDado=TablaIVA.ImporteDado+LinVenta.ImporteDado
        TablaIVA.DescuentoLineaImporte=TablaIVA.DescuentoLineaImporte+
                                                LinVenta.DescuentoLineaImporte
        TablaIVA.DiferenciaIVA=TablaIVA.DiferenciaIVA+LinVenta.DiferenciaIVA
        TablaIVA.DiferenciaEC=TablaIVA.DiferenciaEC+LinVenta.DiferenciaEC
        Modificar registro TablaIVA
    Hasta registro LinVenta sea vacío
CabCompra se inicializa
CabCompra.CambioEmpresa(TablaEmpresa.Nombre)
CabCompra.No=''
CabCompra.TipoDocumento=Factura
Si no inserta registro CabCompra entonces
    ERROR ('Error al crear la cabecera')
CabCompra.Proveedor=1
CabCompra.NoFactura=CabVenta.No
CabCompra.CodigoSubfamilia=4
Modificar registro CabCompra
NumLin=0
TablaIVA se inicializa
Si se posiciona primer registro TablaIVA entonces
    Repetir
        LinCompra se inicializa
        LinCompra se filtra TipoDocumento sea Factura
        LinCompra se filtra NoDocumento sea CabCompra.No
        Si encuentra ultimo registro LinCompra entonces
            NumLin=LinCompra.NoLinea
        LinCompra.CambioEmpresa(TablaEmpresa.Nombre)
        LinCompra se inicializa
        importe=1+(TablaIVA.IVA%/100);
        LinCompra.TipoDocumento=Factura
        LinCompra.NoDocumento=CabCompra.No
        LinCompra.NoLinea=NumLin+1000
        LinCompra.Tipo=Cuenta
        TablaCuenta.Coge('6000001')
        LinCompra.No='6000001'
        LinCompra.GrupoRegProd=TablaCuenta.GrupoRegProd
        LinCompra.GrupoRegNeg=TablaCuenta.GrupoRegNeg
        LinCompra.GrupoRegIVA=TablaCuenta.GrupoRegIVA
```

```
        LinCompra.GrupoImpuesto=TablaCuenta.GrupoImpuesto
        LinCompra.DescuentoFactura=NOT TablaCuenta.DescuentoLinea('6000001')
        LinCompra.AsignacionArticulo=Falso
        LinCompra.CosteMedio=TablaIVA.ImporteLinea
         LinCompra.CosteUnitario(base)=TablaIVA.ImporteLinea
         LinCompra.ImportePendiente=TablaIVA.ImporteLinea*importe
         LinCompra.ImportePendiente(base)=TablaIVA.ImporteLinea*importe
         LinCompra.ImporteLinea=TablaIVA.ImporteLinea
         LinCompra.CosteUnitario=TablaIVA.ImporteLinea
         LinCompra.Descripcion=TablaIVA.IdIVA
         LinCompra.Cantidad=1
         LinCompra.CantidadPendiente=1
         LinCompra.CantidadFacturar=1
         LinCompra.CantidadRecibir=1
         LinCompra.Cantidad(base)=1
         LinCompra.CantidadPendiente(base)=1
         LinCompra.CantidadFacturar(base)=1
         LinCompra.CantidadRecibir(base)=1
         LinCompra.Proveedor=1
         LinCompra.GrupoIVAProd=TablaIVA.IdIVA
         LinCompra.%IVA=TablaIVA.%IVA
         LinCompra.IdIVA=TablaIVA.IdIVA
         LinCompra.CodigoSubfamilia=4
         Insertar registro LinCompra
         TablaDimension.CambiaEmpresa(TablaEmpresa.Nombre)
         TablaDimension se inicializa
         TablaDimension.ID=39
         TablaDimension.TipoDocumento=Factura
         TablaDimension.NoDocumento=CabCompra.No
         TablaDimension.NoLinea=NumLin+1000
         TablaDimension.CodigoValor=4
         Insertar registro TablaDimension
    Hasta registro TablaIVA sea vacío
CabVenta.NoDocumentoExterno=CabCompra.No
Modificar registro CabVenta
```

Código Original

```
rcdEmpresa.RESET;
rcdEmpresa.SETRANGE(rcdEmpresa."Cod. cliente","Sell-to Customer No.");
IF NOT rcdEmpresa.FINDFIRST THEN
  ERROR('No existe la empresa');

rcdauxc.RESET;
rcdauxc.CHANGECOMPANY(rcdEmpresa.Name);
rcdauxc.SETRANGE(rcdauxc."Document Type",rcdauxc."Document Type"::Invoice);
rcdauxc.SETRANGE(rcdauxc."Vendor Invoice No.","No.");
IF rcdauxc.FINDFIRST THEN
  ERROR(' Ya tienes creada la factura de compra\'+
       ' con numero %1\'+
       ' en la empresa %2\',rcdauxc."No.",rcdEmpresa.Name);

rcdauxh.RESET;
rcdauxh.CHANGECOMPANY(rcdEmpresa.Name);
rcdauxh.SETRANGE(rcdauxh."Vendor Invoice No.","No.");
```

```
IF rcdauxh.FINDFIRST THEN
  ERROR(' Ya tienes creada y facturada la factura de compra\'+
       ' con numero %1\'+
       ' en la empresa %2\',rcdauxh."No.",rcdEmpresa.Name);

rcdLin.RESET;
rcdLin.SETRANGE(rcdLin."Document No.","No.");
IF rcdLin.FINDFIRST THEN
BEGIN
   REPEAT
     ven.UPDATE(1,FORMAT('Acumulando importes'));
     IF NOT rcdTemp.GET(rcdLin."VAT Identifier",rcdLin."VAT Calculation
Type",rcdLin."Tax Group Code",FALSE,
                  rcdLin."Line Amount" >=0) THEN
     BEGIN
       rcdTemp.INIT;
       rcdTemp."VAT Identifier":= rcdLin."VAT Identifier";
       rcdTemp."VAT Calculation Type":= rcdLin."VAT Calculation Type";
       rcdVAT.GET(rcdLin."VAT Bus. Posting Group",rcdLin."VAT Prod. Posting Group");
       rcdTemp."EC %":=rcdVAT."EC %";
       rcdTemp."VAT %":=rcdVAT."VAT %";
       rcdTemp.Modified:=TRUE;
       rcdTemp.Positive:=rcdLin."Line Amount" >=0;
       rcdTemp.INSERT;
     END;
     rcdTemp.Quantity:=rcdTemp.Quantity+rcdLin."Quantity (Base)";
     rcdTemp."Line Amount":=rcdTemp."Line Amount"+rcdLin."Line Amount";
      IF rcdLin."Allow Invoice Disc." THEN
        rcdTemp."Inv. Disc. Base Amount":=rcdTemp."Inv. Disc. Base Amount"+rcdLin."Line
Amount";
     rcdTemp."Invoice Discount Amount":=rcdTemp."Invoice Discount Amount"+rcdLin."Inv.
Discount Amount";
     rcdTemp."Pmt. Disc. Given Amount":=rcdTemp."Pmt. Disc. Given Amount"+rcdLin."Pmt.
Disc. Given Amount";
     rcdTemp."Line Discount Amount":=rcdTemp."Line Discount Amount"+rcdLin."Line
Discount Amount";
     rcdTemp."VAT Difference":=rcdTemp."VAT Difference"+rcdLin."VAT Difference";
     rcdTemp."EC Difference":=rcdTemp."EC Difference"+rcdLin."EC Difference";
     rcdTemp.MODIFY;
   UNTIL rcdLin.NEXT=0;
END;

rcdCab.RESET;
rcdCab.CHANGECOMPANY(rcdEmpresa.Name);
rcdCab.INIT;
rcdCab."No.":=";
rcdCab."Document Type":=rcdCab."Document Type"::Invoice;
IF NOT rcdCab.INSERT(TRUE) THEN
  ERROR('Error al crear la cabecera de la factura de compra');

rcdCab.VALIDATE(rcdCab."Buy-from Vendor No.",'1');
rcdCab."Vendor Invoice No.":="No.";
rcdCab.VALIDATE(rcdCab."Shortcut Dimension 1 Code",'4');
rcdCab.MODIFY;
```

```
NumLin:=0;
rcdTemp.RESET;
IF rcdTemp.FINDFIRST THEN
BEGIN
 REPEAT
    ven.UPDATE(1,FORMAT('Creando Factura'));
    rcdLinC.RESET;
    rcdLinC.SETRANGE(rcdLinC."Document Type",rcdLinC."Document Type"::Invoice);
    rcdLinC.SETRANGE(rcdLinC."Document No.",rcdCab."No.");
    IF rcdLinC.FINDLAST THEN
       NumLin:=rcdLinC."Line No.";

    rcdLinC.CHANGECOMPANY(rcdEmpresa.Name);
    rcdLinC.INIT;
    rcdLinC.RESET;
    importe:=1+(rcdTemp."VAT %"/100);
    rcdLinC.VALIDATE(rcdLinC."Document Type",rcdLinC."Document Type"::Invoice);
    rcdLinC.VALIDATE(rcdLinC."Document No.",rcdCab."No.");
    rcdLinC.VALIDATE(rcdLinC."Line No.",NumLin+1000);
    rcdLinC.Type:=rcdLinC.Type::"G/L Account";
    rcdCuenta.GET('6000001');
    rcdCuenta.CheckGLAcc;
    rcdLinC."No.":='6000001';
    rcdLinC."Gen. Prod. Posting Group":=rcdCuenta."Gen. Prod. Posting Group";
    rcdLinC."Gen. Bus. Posting Group":=rcdCuenta."Gen. Bus. Posting Group";
    rcdLinC."VAT Bus. Posting Group":=rcdCuenta."VAT Bus. Posting Group";
    rcdLinC."Tax Group Code":=rcdCuenta."Tax Group Code";
    rcdLinC."Allow Invoice Disc.":=NOT rcdCuenta.InvoiceDiscountAllowed('6000001');
    rcdLinC."Allow Item Charge Assignment":=FALSE;
    rcdLinC."Direct Unit Cost":=rcdTemp."Line Amount";
    rcdLinC."Unit Cost (LCY)":=rcdTemp."Line Amount";
    rcdLinC."Outstanding Amount":=rcdTemp."Line Amount"*importe;
    rcdLinC."Outstanding Amount (LCY)":=rcdTemp."Line Amount"*importe;
    rcdLinC."Line Amount":=rcdTemp."Line Amount";
    rcdLinC."Unit Cost":=rcdTemp."Line Amount";
    rcdLinC.Description:=rcdTemp."VAT Identifier";
    rcdLinC.Quantity:=1;
    rcdLinC."Outstanding Quantity":=1;
    rcdLinC."Qty. to Invoice":=1;
    rcdLinC."Qty. to Receive":=1;
    rcdLinC."Quantity (Base)":=1;
    rcdLinC."Outstanding Qty. (Base)":=1;
    rcdLinC."Qty. to Invoice (Base)":=1;
    rcdLinC."Qty. to Receive (Base)":=1;
    rcdLinC."Pay-to Vendor No.":='1';
    rcdLinC."VAT Prod. Posting Group":=rcdTemp."VAT Identifier";
    rcdLinC."VAT %":=rcdTemp."VAT %";
    rcdLinC."VAT Identifier":=rcdTemp."VAT Identifier";
    rcdLinC."Shortcut Dimension 1 Code":='4';
    rcdLinC.INSERT;
    rcdDIm.CHANGECOMPANY(rcdEmpresa.Name);
    rcdDIm.INIT;
    rcdDIm."Table ID":=39;
```

```
    rcdDIm."Document Type":=rcdDIm."Document Type"::Invoice;
    rcdDIm."Document No.":=rcdCab."No.";
    rcdDIm."Line No.":=NumLin+1000;
    rcdDIm."Dimension Code":='SUBFAMILIA';
    rcdDIm."Dimension Value Code":='4';
    rcdDIm.INSERT;
  UNTIL rcdTemp.NEXT=0;
END;
MESSAGE('FACTURA %1 CREADA\',rcdCab."No.");
"External Document No.":=rcdCab."No.";
MODIFY;
```

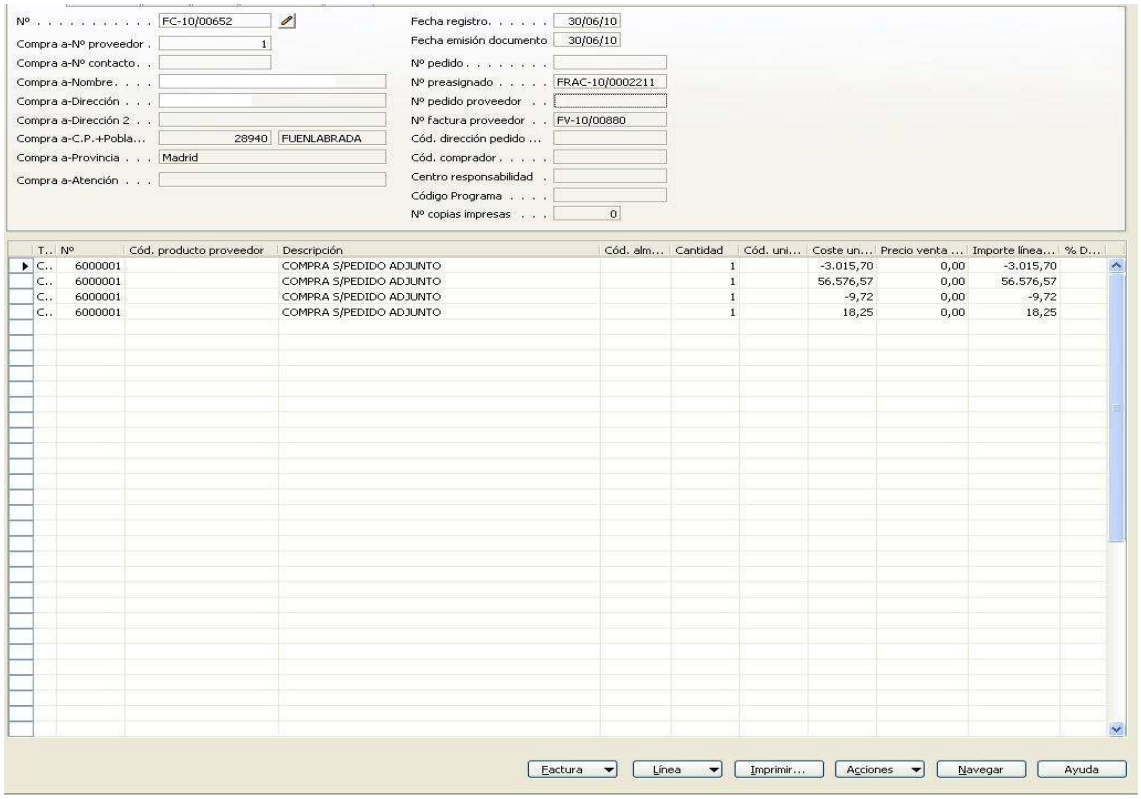


Ilustración 43 Factura Compra que se genera al ejecutar la opción de CrearFacturaCompra

Hoja Precio Venta

En este apartado, si se quiere hacer un cambio masivo de precios de oferta a una serie de productos, existe un formulario donde se puede hacer dicha funcionalidad. En dicho formulario, se debe indicar a que productos se aplican nuevas tarifas. Una vez indicados los productos, para cada uno de ellos se debe rellenar lo siguiente: fecha inicio del nuevo precio, fecha final del nuevo precio, codigo de venta (PVP, Cliente,etc), unidad de medida y nuevo precio venta. Una vez rellenados esos campos, se debe proceder a registrar esos cambios.

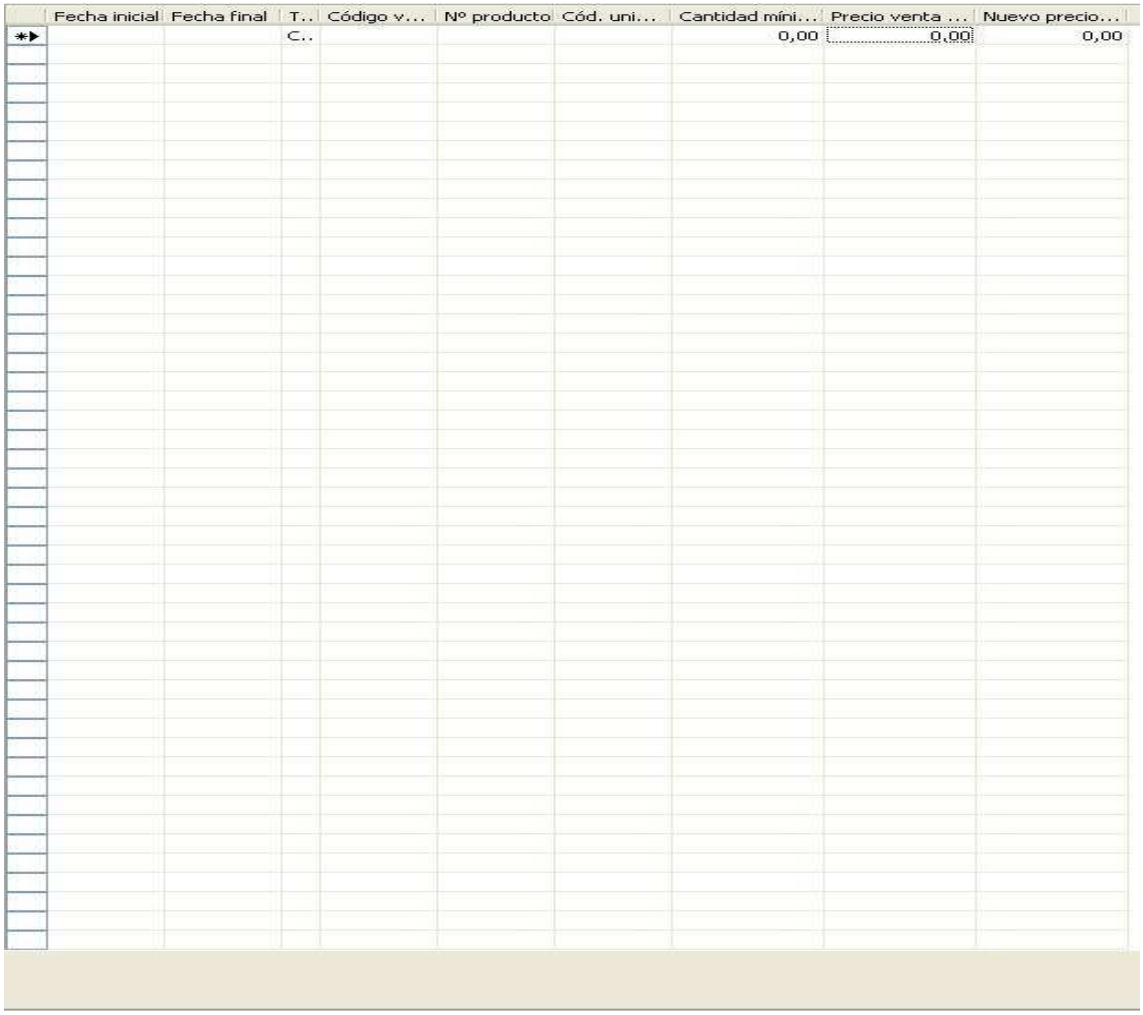


Ilustración 44 Hoja Precio Venta dónde se realizan cambios masivos de precio

PseudoCódigo

HojaVenta se inicializa
Posiciona primer registro HojaVenta
Repetir
  PrecioVenta se inicializa
  PrecioVenta.NoProducto=HojaVenta.NoProducto
  PrecioVenta.TipoVenta=HojaVenta.TipoVenta
  PrecioVenta.CodigoVenta=HojaVenta.CodigoVenta
  PrecioVenta.UnidadMedida=HojaVenta.UnidadMedida
  Precioventa.CodigoVariante=HojaVenta.CodigoVariante
  PrecioVenta.FechaInicio=HojaVenta.FechaInicio
  PrecioVenta.FechaFin=HojaVenta.FechaFin
  PrecioVenta.CantMin=HojaVenta.CantMin
  PrecioVenta.CodigoConcurrencia=HojaVenta. CodigoConcurrencia
  PrecioVenta.PrecioVenta=HojaVenta.PrecioVenta
  PrecioVenta.IncluyeIVA=HojaVenta.IncluyeIVA
  PrecioVenta.DescuentoLinea=HojaVenta.DescuentoLinea
  PrecioVenta.DescuentoFactura=HojaVenta.DescuentoFactura
  PrecioVenta.GrupoPrecioReg=HojaVenta.GrupoPrecioReg
  Si no inserta registro PrecioVenta entonces
    Modificar registro PrecioVenta

Hasta registro HojaVenta sea vacío

Código Original

```
SalesPrice.INIT;
SalesPrice.VALIDATE("Item No.","Item No.");
SalesPrice.VALIDATE("Sales Type","Sales Type");
SalesPrice.VALIDATE("Sales Code","Sales Code");
SalesPrice.VALIDATE("Unit of Measure Code","Unit of Measure Code");
SalesPrice.VALIDATE("Variant Code","Variant Code");
SalesPrice.VALIDATE("Starting Date","Starting Date");
SalesPrice.VALIDATE("Ending Date","Ending Date");
SalesPrice."Minimum Quantity" := "Minimum Quantity";
SalesPrice."Currency Code" := "Currency Code";
SalesPrice."Unit Price" := "New Unit Price";
SalesPrice."Price Includes VAT" := "Price Includes VAT";
SalesPrice."Allow Line Disc." := "Allow Line Disc.";
SalesPrice."Allow Invoice Disc." := "Allow Invoice Disc.";
SalesPrice."VAT Bus. Posting Gr. (Price)" := "VAT Bus. Posting Gr. (Price)";
IF SalesPrice."Unit Price" <> 0 THEN
  IF NOT SalesPrice.INSERT(TRUE) THEN
    SalesPrice.MODIFY(TRUE);
```

4.2.4 Contabilidad

En este apartado se explica cómo se importan asientos en un diario general, cómo se registra un diario de pagos, cómo se generan órdenes de pago para que los bancos puedan procesar el fichero y todo el funcionamiento relacionado con los presupuestos de compra.

Importar Excel(cvSeccion)

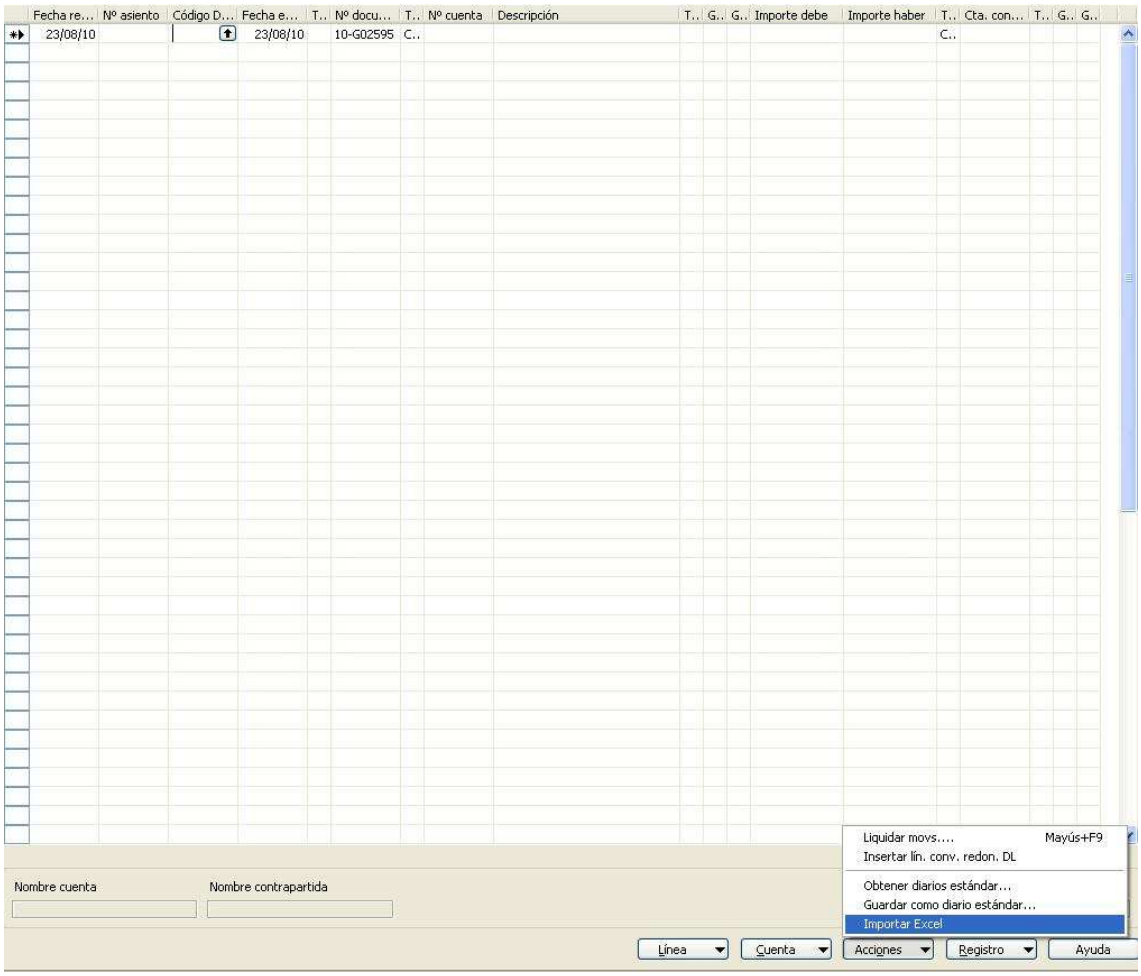


Ilustración 45 Opción de Importar una serie de asientos al diario general

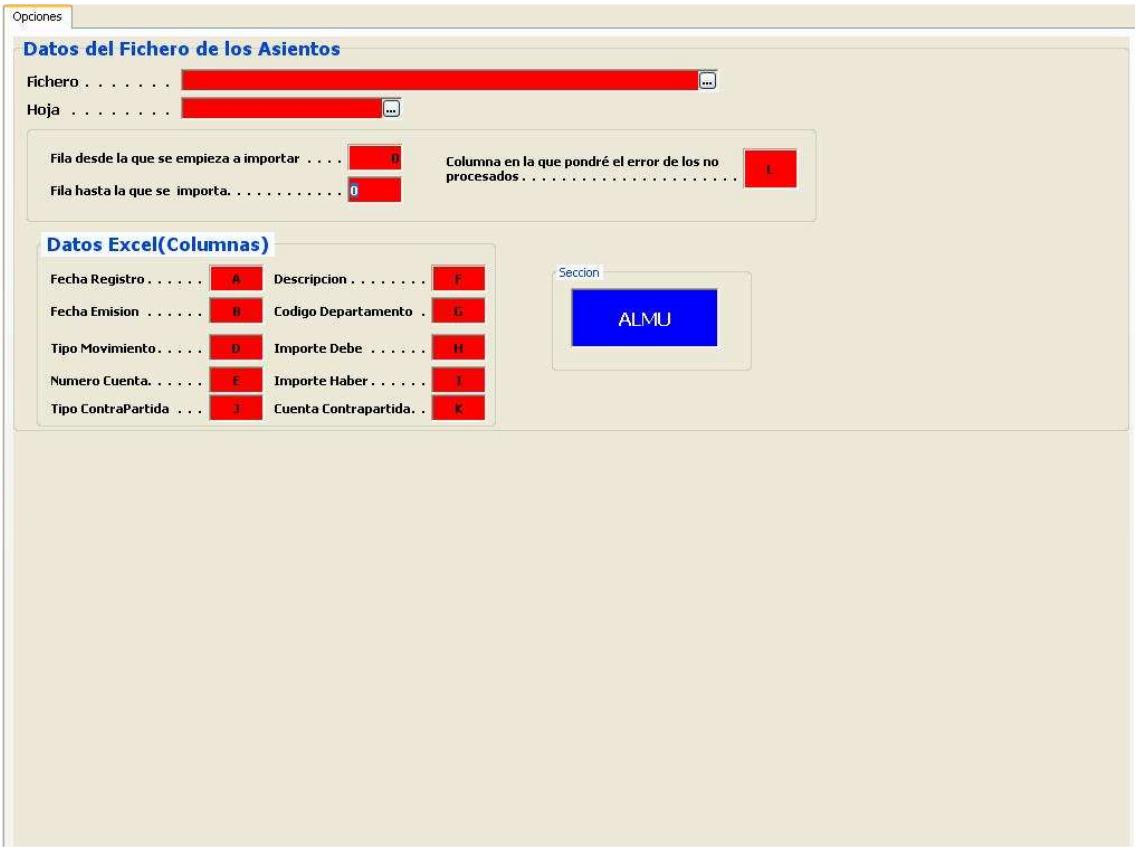


Ilustración 46 Datos que debe rellenar el usuario para que el sistema sepa dónde esta cada dato

En esta funcionalidad, el requisito importante es que el usuario debe rellenar cada uno de los campos tal y como se muestra en la ilustración 46. Una vez rellenados esos campos, el sistema sabe la fecha de registro con la que se quieren registrar los asientos, el tipo de movimiento que se quiere realizar, el número de cuenta dónde se quiere imputar, el importe, etc. En caso de no rellenar algún dato, el sistema emite un mensaje de error indicando que faltan datos. Una vez hechas las validaciones, el sistema comienza a insertar cada uno de los asientos que se encuentran en la hoja de Excel

PseudoCódigo

```
ExcelFechaR=Excell_Dame(Fila,Columna)
QuitaEspacio(ExcelFechaR)
ExcelFechaE=Eliminar(ExcelFechaR, =, .)
ExcelFechaE=Excell_Dame(Fila,Columna)
QuitaEspacio(ExcelFechaE)
ExcelFechaE=Eliminar(ExcelFechaE, =, .)
ExcelTipoMov=Excell_Dame(Fila,Columna)
QuitaEspacio(ExcelTipoMov)
ExcelTipoMov=Eliminar(ExcelTipoMov, =, .)
ExcelNCuenta=Excell_Dame(Fila,Columna)
QuitaEspacio(ExcelNCuenta)
ExcelNCuenta=Eliminar(ExcelNCuenta, =, .)
ExcelDescripcion=Excell_Dame(Fila,Columna)
QuitaEspacio(ExcelDescripcion)
```

```
ExcelDescripcion=Eliminar(ExcelDescripcion, =, .)
ExcelDepartamento=Excell_Dame(Fila,Columna)
QuitaEspacio(ExcelDepartamento)
ExcelDepartamento=Eliminar(ExcelDepartamento, =, .)
ExcelImporteD=Excell_Dame(Fila,Columna)
QuitaEspacio(ExcelImporteD)
ExcelImporteD=Eliminar(ExcelImporteD, =, .)
ExcelImporteH=Excell_Dame(Fila,Columna)
QuitaEspacio(ExcelImporteH)
ExcelImporteH=Eliminar(ExcelImporteH, =, .)
ExcelTipoMovCont=Excell_Dame(Fila,Columna)
QuitaEspacio(ExcelTipoMovCont)
ExcelTipoMovCont =Eliminar(ExcelTipoMovCont, =, .)
ExcelNCuentaCont=Excell_Dame(Fila,Columna)
QuitaEspacio(ExcelNCuentaCont)
ExcelNCuentaCont =Eliminar(ExcelNCuentaCont, =, .)
Si no EVALUA(dvFechaR,ExcelFechaR) entonces
    Excell_Pinta(HojaExcel,Fila,Columna,'Fecha Erronea', Verdadero, Verdadero)
    Saltamos registro
Si no EVALUA(dvFechaE,ExcelFechaE) entonces
    Excell_Pinta(HojaExcel,Fila,Columna,'Fecha Erronea', Verdadero, Verdadero)
    Saltamos registro
NombreSeccion.Coge('GENERAL',cvSeccion)
NumDoc=NumeroSerie(NombreSeccion.NoSeries,dvFechaR)
Si no EVALUA(nvImporteD,ExcelImporteD) entonces
     nvImporteD=0
Si no EVALUA(nvImporteH,ExcelImporteH) entonces
     nvImporteH=0
nvImporteD=Redóndear(nvImporteD,0.01)
nvImporteH=Redóndear(nvImporteH,0.01)
LinDiario se inicializa
LinDiario se filtra NombreDiario sea GENERAL
LinDiario se filtra NombreSeccion sea cvSeccion
Si encuentra registros LinDiario entonces
    nvLin=LinDiario.NoLinea
nvLin=nvLin+10000
LinDiario se inicializa
LinDiario.NombreDiario=GENERAL
LinDiario.NombreSeccion=cvSeccion
LinDiario.NoLinea=nvLin
LinDario.NoDocumento=NumDoc
LinDiario.CodigoOrigen='DIAGEN'
LinDiario.FechaRegistro=dvFechaR
LinDiario.FechaEmision=dvFechaE
Si ExcelTipoMov=cuenta entonces
   LinDiario.TipoCuenta=Cuenta
Si ExcelTipoMov=cliente entonces
   LinDiario.TipoCuenta=Cliente
Si ExcelTipoMov=proveedor entonces
   LinDiario.TipoCuenta=Proveedor
Si ExcelTipoMov=banco entonces
   LinDiario.TipoCuenta=Banco
Si ExcelTipoMov=activo entonces
   LinDiario.TipoCuenta=Activo
```

```
Si ExcelTipoMov=empresavinculadaasociada entonces
   LinDiario.TipoCuenta=Empresavinculadaasociada
LinDiario.NoCuenta=ExcelNCuenta
Si ExcelTipoMovCont=cuenta entonces
   LinDiario.TipoCuentaCont=Cuenta
Si ExcelTipoMovCont=cliente entonces
   LinDiario.TipoCuentaCont=Cliente
Si ExcelTipoMovCont=proveedor entonces
   LinDiario.TipoCuentaCont=Proveedor
Si ExcelTipoMovCont=banco entonces
   LinDiario.TipoCuentaCont=Banco
Si ExcelTipoMovCont=activo entonces
   LinDiario.TipoCuentaCont=Activo
Si ExcelTipoMovCont=empresavinculadaasociada entonces
   LinDiario.TipoCuentaCont=Empresavinculadaasociada
Si ExcelNCuentaCont<>'' entonces
   LinDiario.NoCuentaCont=ExcelNCuentaCont
LinDiario.Descripcion=ExcelDescripcion
Si nvImporteD<>0 entonces
   LinDiario.ImporteDebe=nvImporteD
Si nvImporteH<>0 entonces
   LinDiario.ImporteHaber=nvImporteH
LinDiario.CodigoDepartamento=ExcelDepartamento
Insertar Registro LinDiario
```

Código Original

```
ExcelFechaR:=Excell_Dame(i,Excell_ColChar_a_Num(optColFechR));
QuitaEspacio(ExcelFechaR);
// Limpiamos el código de puntos
ExcelFechaR:=DELCHR(ExcelFechaR,'=','.');

ExcelFechaE:=Excell_Dame(i,Excell_ColChar_a_Num(optColFechE));
QuitaEspacio(ExcelFechaE);
// Limpiamos el código de puntos
ExcelFechaE:=DELCHR(ExcelFechaE,'=','.');

ExcelTipoMov:=Excell_Dame(i,Excell_ColChar_a_Num(optColTipoMov));
QuitaEspacio(ExcelTipoMov);
// Limpiamos el código de puntos
ExcelTipoMov:=DELCHR(ExcelTipoMov,'=','.');

ExcelNCuenta:=Excell_Dame(i,Excell_ColChar_a_Num(optColNCuenta));
QuitaEspacio(ExcelNCuenta);
// Limpiamos el código de puntos
ExcelNCuenta:=DELCHR(ExcelNCuenta,'=','.');

ExcelDescripcion:=Excell_Dame(i,Excell_ColChar_a_Num(optColDescripcion));
QuitaEspacio(ExcelDescripcion);
// Limpiamos el código de puntos
ExcelDescripcion:=DELCHR(ExcelDescripcion,'=','.');

ExcelDepartamento:=Excell_Dame(i,Excell_ColChar_a_Num(optColDepartamento));
QuitaEspacio(ExcelDepartamento);
// Limpiamos el código de puntos
```

```
ExcelDepartamento:=DELCHR(ExcelDepartamento,'=','.');

ExcelImporteD:=Excell_Dame(i,Excell_ColChar_a_Num(optColImporteD));
QuitaEspacio(ExcelImporteD);
// Limpiamos el código de puntos
ExcelImporteD:=DELCHR(ExcelImporteD,'=','.');

ExcelImporteH:=Excell_Dame(i,Excell_ColChar_a_Num(optColImporteH));
QuitaEspacio(ExcelImporteH);
// Limpiamos el código de puntos
ExcelImporteH:=DELCHR(ExcelImporteH,'=','.');

ExcelTipoMovCont:=Excell_Dame(i,Excell_ColChar_a_Num(optColTipoCont));
QuitaEspacio(ExcelTipoMovCont);
// Limpiamos el código de puntos
ExcelTipoMovCont:=DELCHR(ExcelTipoMovCont,'=','.');

ExcelNCuentaCont:=Excell_Dame(i,Excell_ColChar_a_Num(optColNCuentaCont));
QuitaEspacio(ExcelNCuentaCont);
// Limpiamos el código de puntos
ExcelNCuentaCont:=DELCHR(ExcelNCuentaCont,'=','.');

IF NOT EVALUATE(dvFechaR,ExcelFechaR) THEN
BEGIN
   Excell_Pinta(HojaExcel,i,Excell_ColChar_a_Num(optColError),'Fecha                Registro
erronea',TRUE,TRUE);
   nvNoProcesados+=1;
   Ven.UPDATE(3,nvNoProcesados);
   CurrReport.SKIP;
END;

IF NOT EVALUATE(dvFechaE,ExcelFechaE) THEN
BEGIN
   Excell_Pinta(HojaExcel,i,Excell_ColChar_a_Num(optColError),'Fecha                Emision
erronea',TRUE,TRUE);
   nvNoProcesados+=1;
   Ven.UPDATE(3,nvNoProcesados);
   CurrReport.SKIP;
END;
COMMIT;

CLEAR(NoSeriesMgt);
GenJnlBatch.GET('GENERAL',cvSeccion);
NumDoc:=NoSeriesMgt.TryGetNextNo(GenJnlBatch."No. Series",dvFechaR);

IF NOT EVALUATE(nvImporteD,ExcelImporteD) THEN
  nvImporteD:=0;

IF NOT EVALUATE(nvImporteH,ExcelImporteH) THEN
  nvImporteH:=0;

nvImporteD:=ROUND(nvImporteD,0.01);
nvImporteH:=ROUND(nvImporteH,0.01);
```

```
rcdDiario.RESET;
rcdDiario.SETRANGE(rcdDiario."Journal Template Name",'GENERAL');
rcdDiario.SETRANGE(rcdDiario."Journal Batch Name",cvSeccion);
IF rcdDiario.FINDLAST THEN
   nvLin:=rcdDiario."Line No.";

nvLin:=nvLin+10000;
rcdDiario.INIT;
rcdDiario."Journal Template Name":='GENERAL';
rcdDiario."Journal Batch Name":=cvSeccion;
rcdDiario."Line No.":=nvLin;
rcdDiario."Document No.":=NumDoc;
rcdDiario."Source Code":='DIAGEN';
rcdDiario.VALIDATE(rcdDiario."Posting Date",dvFechaR);
rcdDiario.VALIDATE(rcdDiario."Document Date",dvFechaE);
IF ExcelTipoMov='cuenta' THEN
  rcdDiario."Account Type":=rcdDiario."Account Type"::"G/L Account";

IF ExcelTipoMov='cliente' THEN
  rcdDiario."Account Type":=rcdDiario."Account Type"::Customer;

IF ExcelTipoMov='proveedor' THEN
  rcdDiario."Account Type":=rcdDiario."Account Type"::Vendor;

IF ExcelTipoMov='banco' THEN
  rcdDiario."Account Type":=rcdDiario."Account Type"::"Bank Account";

IF ExcelTipoMov='activo' THEN
  rcdDiario."Account Type":=rcdDiario."Account Type"::"Fixed Asset";

IF ExcelTipoMov='empresavinculadaasociada' THEN
  rcdDiario."Account Type":=rcdDiario."Account Type"::"IC Partner";

rcdDiario.VALIDATE(rcdDiario."Account No.",ExcelNCuenta);

IF ExcelTipoMovCont='cuenta' THEN
   rcdDiario."Bal. Account Type":=rcdDiario."Bal. Account Type"::"G/L Account";

IF ExcelTipoMovCont='cliente' THEN
   rcdDiario."Bal. Account Type":=rcdDiario."Bal. Account Type"::Customer;

IF ExcelTipoMovCont='proveedor' THEN
  rcdDiario."Bal. Account Type":=rcdDiario."Bal. Account Type"::Vendor;

IF ExcelTipoMovCont='banco' THEN
  rcdDiario."Bal. Account Type":=rcdDiario."Bal. Account Type"::"Bank Account";

IF ExcelTipoMovCont='activo' THEN
  rcdDiario."Bal. Account Type":=rcdDiario."Bal. Account Type"::"Fixed Asset";

IF ExcelTipoMovCont='empresavinculadaasociada' THEN
  rcdDiario."Bal. Account Type":=rcdDiario."Bal. Account Type"::"IC Partner";

IF ExcelNCuentaCont<>'' THEN
```

```
  rcdDiario.VALIDATE(rcdDiario."Bal. Account No.",ExcelNCuentaCont);
rcdDiario.Description:=ExcelDescripcion;
IF nvImporteD<>0 THEN
  rcdDiario.VALIDATE(rcdDiario."Debit Amount",nvImporteD);
IF nvImporteH<>0 THEN
  rcdDiario.VALIDATE(rcdDiario."Credit Amount",nvImporteH);
rcdDiario.VALIDATE(rcdDiario."Shortcut Dimension 2 Code",ExcelDepartamento);
rcdDiario.INSERT(TRUE);
```

Control Diario Pagos(LinDiario)

El control del diario de pagos consiste en comprobar si el pago que se va a registrar se puede imputar en el mes correspondiente o por el contrario no existe presupuesto suficiente para afrontar el pago. También se puede dar el caso de que se quiera reclasificar una serie de pagos, es decir, imputar esos pagos a otro mes porque en el mes que corresponda no se pueden afrontar. En cualquiera de los dos casos, el sistema genera unos registros detallados dando los documentos como pagados.

PseudoCódigo

```
Si LinDiario.NombreDiario<>'' Y LinDiario.NombreSeccion<>'' entonces
    LinDiario se filtra NombreDiario sea LinDiario.NombreDiario
    LinDiario se filtra NombreSeccion sea LinDiario.NombreSeccion
    Si ConfirmarPptosDiario(LinDiario) entonces
        ActualizarPptosDiario(LinDiario)
    Sino
        ERROR ('Proceso Cancelado')
```

Código Original

```
IF ((GenJnlLine."Journal Template Name"<>'') AND (GenJnlLine."Journal Batch Name"<>'')) THEN
BEGIN
  GenJnlLine.SETRANGE(GenJnlLine."Journal Template Name",GenJnlLine."Journal Template
Name");
  GenJnlLine.SETRANGE(GenJnlLine."Journal Batch Name",GenJnlLine."Journal Batch Name");
  IF ConfirmarPptosDiario(GenJnlLine) THEN
    ActualizaPptosDiario(GenJnlLine)
  ELSE
    ERROR('Registro cancelado por el usuario');
END;
```

ConfirmarPptosDiarios(LinDiario)

PseudoCódigo

```
LinDiario se posiciona en el primer registro
Repetir
    Si LinDiario.TipoCuenta=Proveedor Y LinDiario.NoDocAplicar<>''
        DetallePrespuesto se inicializa
        DetallePresupuesto se filtra NoDocumento sea LinDiaro.NoDocAplicar
        DetallePresupuesto se filtra NoEfecto sea LinDiario.NoFactura
        DetallePresupuesto se filtra Tipo sea Cartera
        Si encuentra registros DetallePresupuesto entonces
            Si longitud(cvMsg)<950 entonces
```

```
            cvMsg+= '\'+LinDiario.NoDocAplicar+' – '+LinDiario.NoFactura+
                    DetallePresupuesto.CodFamilia+' – '+
                    DetallePresupuesto.NoPptoMes+ ' – '
          Si longitud(cvMsg)<950 entonces
            cvMsg+='Pago de'+ LinDiario.Importe
          LinDiario.Subfamilia=DetallePresupuesto.CodFamilia
          Modificar registro LinDiario
Hasta registro LinDiario sea vacío
Si cvMsg<>'' entonces
   Si cvMsg>=950 entonces
      cvMsg+='\.....no caben mas';
   SALIR(CONFIRMAR('-Movimientos Implicados en los presupuestos-\'+cvMsg,
         FALSE,cvMsg));
SALIR(TRUE);
```

Código Original

```
GenJnlLine.FINDSET();
REPEAT
 IF (GenJnlLine."Account Type"=GenJnlLine."Account Type"::Vendor) AND
(GenJnlLine."Applies-to Doc. No." <> '') THEN
 BEGIN
  rcdDetallePpto.RESET;
  rcdDetallePpto.SETCURRENTKEY("Nº Ppto Año","Nº Documento","Nº Efecto");
  rcdDetallePpto.SETRANGE("Nº Documento",GenJnlLine."Applies-to Doc. No.");
  rcdDetallePpto.SETRANGE("Nº Efecto",GenJnlLine."Applies-to Bill No.");
  rcdDetallePpto.SETRANGE(Tipo,rcdDetallePpto.Tipo::Cartera);
  IF rcdDetallePpto.FINDFIRST() THEN
  BEGIN

    IF STRLEN(cvMsg) <950 THEN
      cvMsg+= '\'+GenJnlLine."Applies-to Doc. No." +' - ' +
               GenJnlLine."Applies-to Bill No." + ' - ' +
               rcdDetallePpto."Cód. Familia Presupuestaria" + ' - '+
               FORMAT(rcdDetallePpto."Nº Ppto Mes") + ' - ';

    IF STRLEN(cvMsg) <950 THEN
       cvMsg+='Pago de ' + FORMAT(GenJnlLine.Amount);

   GenJnlLine.VALIDATE(GenJnlLine."Shortcut Dimension 1 Code",rcdDetallePpto."Cód.
Familia Presupuestaria");
    GenJnlLine.MODIFY;
  END;
 END;
UNTIL GenJnlLine.NEXT=0;

IF cvMsg<>'' THEN
BEGIN
  IF STRLEN(cvMsg) >= 950 THEN
    cvMsg+='\..... no caben más';

  EXIT(CONFIRM('- MOVIMIENTOS IMPLICADOS EN LOS PRESUPUESTOS DE
COMPRA -\'+cvMsg,FALSE,cvMsg));

END;
```

```
EXIT(TRUE);
```

ActualizarPptosDiario(LinDiario)

PseudoCódigo

```
LinDiario se posiciona en el primer registro
Repetir
       Si LinDiario.TipoCuenta=Proveedor Y LinDiario.NoDocAplicar<>''
        DetallePrespuesto se inicializa
        DetallePresupuesto se filtra NoDocumento sea LinDiaro.NoDocAplicar
        DetallePresupuesto se filtra NoEfecto sea LinDiario.NoFactura
        DetallePresupuesto se filtra Tipo sea Cartera
        Si encuentra registros DetallePresupuesto entonces
           Si no LinDiario.ReclasDeuda entonces
              GrabaDetalle(LinDiario.Subfamilia, DetallePresupuesto.FechVcto,
              LinDiario.Importe,Pagado,LinDiario.NoDocAplicar, LinDiario.NoFactura,
              LinDiario.NoCuenta,0D,LinDiario.Subfamilia,Falso, LinDiario.Subfamilia)
          Si DetallePresupuesto.Importe=LinDiario.Importe entonces
             Borrar DetallePresupuesto
          Sino
             DetallePresupuesto.Importe-=LinDiario.Importe
             Modificar registro DetallePresupuesto
        Si  LinDiario.TipoCuenta=Proveedor Y LinDiario.NoDocAplicar<>'' Y
            LinDiario.ReclasDeuda entonces
               GrabaDetalle(LinDiario.Subfamilia, DetallePresupuesto.FechVcto,
              -1*LinDiario.Importe,Cartera,LinDiario.NoDoc,
              LinDiario.NoFactura,LinDiario.NoCuenta,0D,LinDiario.Subfamilia,Falso,
              LinDiario.Subfamilia)
Hasta que registro LinDiario sea vacío
```

Código Original

```
GenJnlLine.FINDSET();
REPEAT
 IF (GenJnlLine."Account Type"=GenJnlLine."Account Type"::Vendor) AND
(GenJnlLine."Applies-to Doc. No." <> '') THEN
 BEGIN
  rcdDetallePpto.RESET;
  rcdDetallePpto.SETCURRENTKEY("Nº Ppto Año","Nº Documento","Nº Efecto");
  rcdDetallePpto.SETRANGE("Nº Documento",GenJnlLine."Applies-to Doc. No.");
  rcdDetallePpto.SETRANGE("Nº Efecto",GenJnlLine."Applies-to Bill No.");
  rcdDetallePpto.SETRANGE(Tipo,rcdDetallePpto.Tipo::Cartera);
  IF rcdDetallePpto.FINDFIRST() THEN
  BEGIN
    IF NOT GenJnlLine."Reclas. Deuda" THEN
      GrabaDetalle(GenJnlLine."Shortcut Dimension 1 Code",rcdDetallePpto."Fecha
Vencimiento",GenJnlLine.Amount,
                optTipo::Pagado,GenJnlLine."Applies-to Doc. No.",GenJnlLine."Applies-to Bill
No.",
                GenJnlLine."Account No.",0D,GenJnlLine."Shortcut Dimension 1
Code",FALSE,
                GenJnlLine."Shortcut Dimension 1 Code");

    IF rcdDetallePpto.Importe = GenJnlLine.Amount THEN
      rcdDetallePpto.DELETE
```

```
      ELSE
      BEGIN
        rcdDetallePpto.Importe-= GenJnlLine.Amount;
        rcdDetallePpto.MODIFY;
      END;
    END;
  END;

  IF (GenJnlLine."Account Type"=GenJnlLine."Account Type"::Vendor) AND
(GenJnlLine."Applies-to Doc. No." =") AND
    (GenJnlLine."Reclas. Deuda") THEN
  BEGIN
        ComprobarPresupDiario(GenJnlLine);
        GrabaDetalle(GenJnlLine."Shortcut Dimension 1 Code",GenJnlLine."Due Date",-
1*GenJnlLine.Amount,
                optTipo::Cartera,GenJnlLine."Document No.",GenJnlLine."Bill No.",
                GenJnlLine."Account No.",0D,GenJnlLine."Shortcut Dimension 1
Code",FALSE,
                GenJnlLine."Shortcut Dimension 1 Code");
  END;
UNTIL GenJnlLine.NEXT=0;
```

ComprobarPresupDiario(LinDiario)

PseudoCódigo

```
ConfigEmpresa.Coge
TablaDimension.Coge(ConfigEmpresa.Subfamilia,LinDiario.CodigoSubfamilia)
PresupuestoMes se filtra TipoPresupuesto sea TablaDimension.TipoPresupuesto
PresupuestoMes se filtra Fecha Inicial sea >0D y <=LinDiario.FechaVcto
PresupuestoMes se filtra Fecha Final sea >LinDiario.FechaVcto y <=31/12/9999
Si no encuentra registros PresupuestoMes entonces
   ERROR( ‘ No existe ningun presupuesto’);
PresupuestoFamilia se filtra TipoPresupuesto sea TablaDimension.TipoPresupuesto
PresupuestoFamilia se NoPptoAño sea PrespuestoMes.NoPptoAño
PresupuestoFamilia se NoPptoMes sea PrespuestoMes.NoPptoMes
PresupuestoFamilia se filtra CodFamilia sea LinDiario.CodigoSubfamilia
Si no encuentra registros PresupuestoFamilia entonces
    ERROR (‘No existe ningun presupuesto)
PresupuestoFamilia.CalculaCampos(ImporteConsumidoTotal)
Si PresupuestoFamilia.ImporteConsumidoTotal+(-1*LinDiario.Importe) >
   PresupuestoFamilia.ImportePresupuesto entonces
       ERROR (‘Presupuesto Sobrepasado’);
```

Código Original

```
GLSetup.GET;
DimVal.GET(GLSetup."Global Dimension 1 Code",LinDiario2."Shortcut Dimension 1 Code");
rcdPresupMes.SETRANGE(rcdPresupMes."Tipo Presupuesto",DimVal."Tipo Presupuesto");
rcdPresupMes.SETRANGE("Fecha Inicial",0D,LinDiario2."Due Date");
rcdPresupMes.SETRANGE("fecha final",LinDiario2."Due Date",31129999D);

IF NOT rcdPresupMes.FINDFIRST() THEN
  ERROR('No existe ningún presupuesto creado para %1',FORMAT(LinDiario2."Due Date"));
```

```
rcdPresupFamilia.SETRANGE(rcdPresupFamilia."Tipo Presupuesto",DimVal."Tipo
Presupuesto");
rcdPresupFamilia.SETRANGE("Nº Ppto Año",rcdPresupMes."Nº Ppto Año");
rcdPresupFamilia.SETRANGE("Nº Ppto Mes",rcdPresupMes."Nº Ppto Mes");
rcdPresupFamilia.SETRANGE("Cód. Familia
Presupuestaria",FunPPTO.DevDimValorPresup(LinDiario2."Shortcut Dimension 1 Code"));
IF NOT rcdPresupFamilia.FINDFIRST() THEN
  MESSAGE('No existe ningún presupuesto creado para %1 - %2 - %3',
       FORMAT(rcdPresupMes."Nº Ppto Año"),FORMAT(rcdPresupMes."Nº Ppto Mes"),
       FunPPTO.DevDimValorPresup(LinDiario2."Shortcut Dimension 1 Code"));

rcdPresupFamilia.CALCFIELDS(rcdPresupFamilia."Importe Consumido Total");
IF rcdPresupFamilia."Importe Consumido Total"+(-1*LinDiario2.Amount) >
rcdPresupFamilia."Importe Presupuestado" THEN
  ERROR('¡¡¡¡¡ ATENCION !!!!!!!\\'+
      'PRESUPUESTO SOBREPASADO\'+
      '--------------------------------------\'+
      '\Documento ' + FORMAT(LinDiario2."Document No.") + 'Efecto
'+FORMAT(LinDiario2."Bill No.")+
      '\Presupuesto ' + FORMAT(rcdPresupFamilia."Nº Ppto Año") +
      '\Mes ' + FORMAT(rcdPresupMes."Descripción Periodo") +
      '\Familia ' + FORMAT(rcdPresupFamilia."Cód. Familia Presupuestaria") +
      '\Importe Presupuestado ' + FORMAT(rcdPresupFamilia."Importe Presupuestado") +
      '\Importe Consumido ' + FORMAT(rcdPresupFamilia."Importe Consumido Total") +
      '\Importe Vencimiento ' + FORMAT(-1*LinDiario2.Amount));
// +001
```

Órdenes de Pago

En esta funcionalidad, lo que se quiere es facilitar el trabajo al banco para cuando se le entregue una serie de pagos. Para ello se hace lo siguiente:

En primer lugar, se debe generar una nueva orden de pago. Una vez generada, se deben insertar los registros que se quieren incluir en la orden de pago, para que posteriormente pueda procesarlo el banco. Una vez incluidos, existe la opción de generar la Norma68 y generar un pago único de todos los registros, así el banco puede codificarlo de manera sencilla.

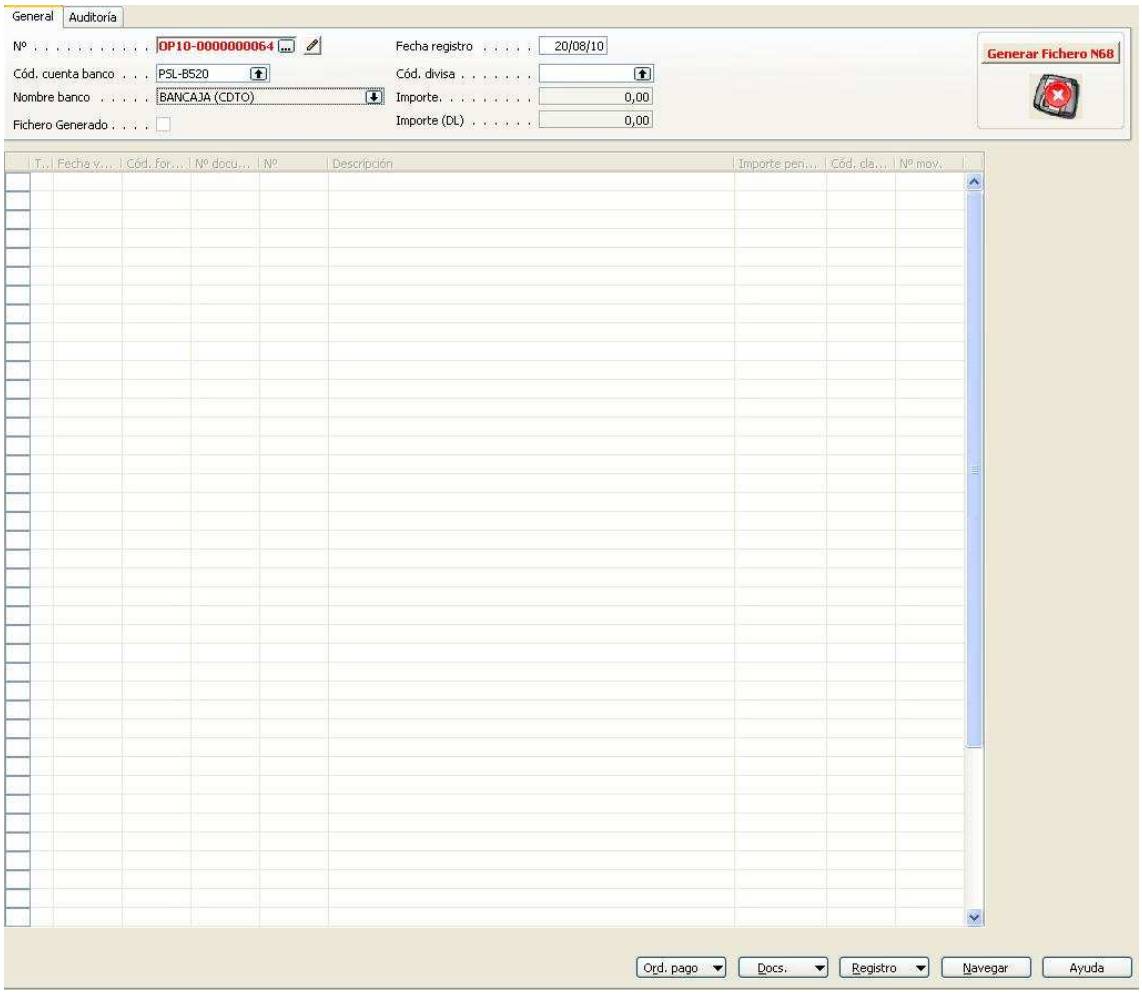


Ilustración 47 Orden de Pago Generada dónde tiene habilitada la opción de generar Norma 68

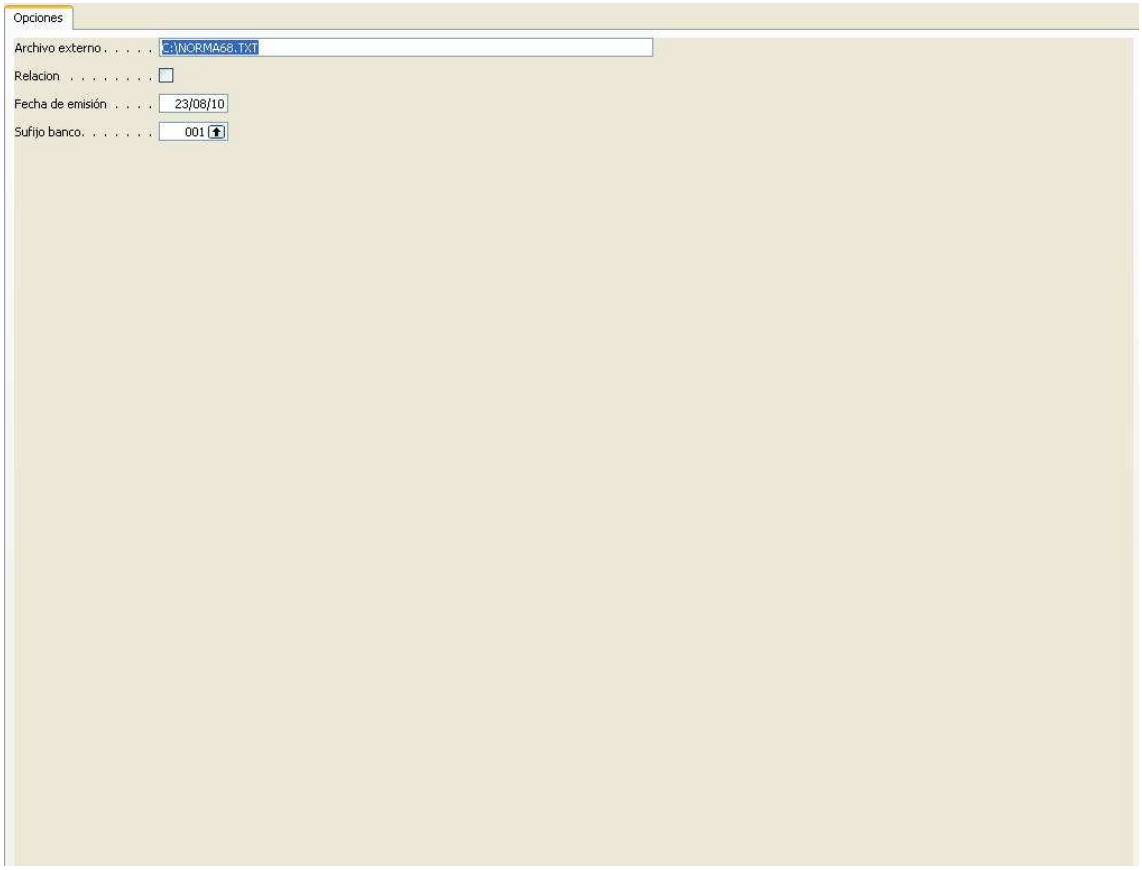


Ilustración 48 Condiciones que se deben indicar para generar el fichero de la norma 68

Norma68

PseudoCódigo

```
ConfigEmpresa.Coge
Si TieneErrores entonces
   Relat='1'
Sino
   Relat='0'
OrdenPago se posiciona en el primer registro
EsEuro=CodigoRegistro(OrdenPago.CodigoConcurrencia,CodigoRegistro,
                      CadenaRegistro)
Si CodigoRegistro<>0 entonces
    CadenaRegistro='56'
Sino
   CadenaRegistro='06'

Tablabanco.Coge(OrdenPago.NoCuentaBanco)
CuentaBanco=Tablabanco.NoBanco
CuentaBanco=TRUNCAR('',MaximaLongitud(CuentaBanco)-Longitud(CuentaBanco),
                    '0')+CuentaBanco
CuentaSucursal=Tablabanco.NoSucursal
CuentaSucursal=TRUNCAR('',MaximaLongitud(CuentaSucursal)-
                      Longitud(CuentaSucursal),'0')+CuentaSucursal
CuentaBanco=Tablabanco.NoCuentaBanco
NoCuentaBanco=TRUNCAR('',MaximaLongitud(NoCuentaBanco)-
                      Longitud(NoCuentaBanco), '0')+NoCuentaBanco
```

```
ControlDigitos=Tablabanco.ControlDigitos
ControlDigitos =TRUNCAR(‘’,MaximaLongitud(ControlDigitos)-
                                Longitud(ControlDigitos),‘0’)+ ControlDigitos
IBAN=CopiarCadena(Tablabanco.IBAN,1,4)
Si DocCartera.FechaRegistro=0D entonces
     FechRegistro=TRUNCAR(‘’,6,’0’)
Sino
    FechRegistro=Formateo(OrdenPago.FechaRegistro,0, <Day,2><month,2><year>)
TextoFichero=’03’+’59’+TRUNCAR(OrdenPago.RegistroIVA,9,’ ‘)+SufijoBanco+
                TRUNCAR(‘’,12,’ ‘)+’001’+Formateo(OrdenPago.FechaEntrega,0,
                <Day,2><month,2><year>)+TRUNCAR(‘’,9.’  ‘)+
                TRUNCAR(IBAN,4,’’)+CuentaBanco+CuentaSucursal+ControlDigitos
                NoCuentaBanco+TRUNCAR(‘’,30.’ ‘);
FicheroTexto.Escribir(TextoFichero)
TotalRegistros=TotalRegistros+1
ProveedorAnt=’’
VctoAnt=0D
OrdenPago.FicheroGenerado=Verdadero
Modificar registro OrdenPago
nvLin=0
TablaComentario se inicializa
TablaComentario se filtra No sea OrdenPago.No
TablaComentario se filtra Tipo sea Pago
Si encuentra ultimo registro entonces
   nvLin=TablaComentario.NoLinea
nvLin=nvLin+10000
TablaComentario se inicializa
TablaComentario.No=OrdenPago.No
TablaComentario.Tipo=Pago
TablaComentario.NoLinea=nvLin
TablaComentario.Fecha=Hoy
TablaComentario.Comentario=Formateo(Hoy)+’ – ‘+Usuario+’ – ‘+Generacion de
                                Fichero’+FicheroTexto+’.’;
Insertar registro TablaComentario
TablaProveedor.Coge(DocCartera.NoCuenta)
RegProvIVA=TablaProveedor.RegistroIVA
RegProvIVA=TRUNCAR(‘’,Maxlongitud(RegProvIVA)-Longitud(RegProvIVA),’ ‘)+
              RegProvIVA
Importe=TRUNCAR(‘’,Maxlongitud(Importe)-Longitud(Importe),’0 ‘)+Importe
Si No TablaPais.Coge(TablaProveedor.CodigoPais) entonces
    TablaPais se inicializa
    TablaPais.Code=’’
Caso Verdadero
  Moneda=Peseta: Si esEuro entonces
                    TotalImporte=TotalImporte+DocCartera.ImporteRestante
                    ImporteDocumento=ImporteRestante
                    Importe=EuroImporte(ImporteDocumento)
                 Sino
                    Importe=Formateo(ImporteRestante*100,12, <Integer>)
                    TotalImporte=TotalImporte+DocCartera.ImporteRestante
  Moneda=Euro:   Si esEuro entonces
                    TotalImporte=TotalImporte+DocCartera.ImporteRestante
                    ImporteDocumento=ImporteRestante
                    Importe=EuroImporte(ImporteDocumento)
```

```
                    Sino
                       Importe=Formateo(ImporteRestante*100,12, <Integer>)
                       TotalImporte=TotalImporte+DocCartera.ImporteRestante

Moneda=Otro:        Si esEuro entonces
                       TotalImporte=TotalImporte+DocCartera.ImporteRestante
                       ImporteDocumento=ImporteRestante
                       Importe=EuroImporte(ImporteDocumento)
                    Sino
                       Importe=Formateo(ImporteRestante*100,12, <Integer>)
                       TotalImporte=TotalImporte+DocCartera.ImporteRestante
TextoFichero='06'+'59'+Truncar(RegNoIVA,9,' ')+SufijoBanco+RegProvIVA+'010'
              +Truncar(Proveedor.Nombre,40,' ')+Truncar('',29,' ');
FicheroTexto.Escribir(TextoFichero)
TotalReg=TotalReg+1
TotalDocProv=TotalDocProv+1
TextoFichero='06'+'59'+Truncar(RegNoIVA,9,' ')+SufijoBanco+RegProvIVA+'011'
              +Truncar(Proveedor.Direccion,45,' ')+Truncar('',24,' ');
FicheroTexto.Escribir(TextoFichero)
TotalReg=TotalReg+1
TextoFichero='06'+'59'+Truncar(RegNoIVA,9,' ')+SufijoBanco+RegProvIVA+'012'
              +Truncar(Proveedor.CodigoPost,5,' ')+Truncar(Proveedor.Ciudad,40,' ')
              +Truncar('',24,' ');
FicheroTexto.Escribir(TextoFichero)
TotalReg=TotalReg+1
TextoFichero='06'+'59'+Truncar(RegNoIVA,9,' ')+SufijoBanco+RegProvIVA+'010'
              +Truncar(Proveedor.CodigoPost,9,' ')+Truncar(Proveedor.Pais,30,' ')
             +Truncar(Pais.Nombre,20,' ')+Truncar('',10,' ');
FicheroTexto.Escribir(TextoFichero)
TotalReg=TotalReg+1
Tablabanco.UltimoNoCheque=Incrementar(TablaBanco.UltimoNoCheque)
Modificar registro Tablabanco
NumPagare=Tablabanco.UltimoNoCheque
NumPagare=Truncar('',Maxlong(NumPagare)-longitud(NumPagare),0)+NumPagare
Evaluar(Dividendo,'9000'+NumPagare)
DigitoControl=DividendoMOD7
NumPago=NumPagare+Formateo(DigitoControl)
ProvAnt=TablaProveedor.No
VctoAnt=DocCartera.FechaVencimiento
TextoFichero=''06'+'59'+Truncar(NoRegIVA,9,'')+SufijoBanco+RegProvIVA+'014'
              +NumPago+Formateo(DocCartera.FechaVcto,0,
              <Day,2><month,2><year4>)+Convertir(Importe,' ,'0')+'0'+
              Truncar('',2,' ')+Truncar('',6,' ')+Truncar('',31,' ')+'1'
FicheroTexto.Escribir(TextoFichero)
TotalReg=TotalReg+1
Concepto=' '
NoDoc=''
Si MovProducto.Coge(DocCartera.NoMov) entonces
    Concepto=CopiarCadena(MovProducto.NoDocExterno,1,26)
    NoDoc=CopiarCadena(MovProducto.NoDocExterno,1,12)
Si Concepto='' entonces
   Concepto=CopiarCadena(DocCartera.Descripcion,1,26)
TextoFichero='06'+'59'+Truncar(NoRegIVA,9,' ')+SufijoBanco+RegProvIVA+'015'
              +NumPago+Truncar(NoDoc,12,'')+ Formateo(FechaEntrega,0,
```

```
                <Day,2><month,2><year4>)+ConvertirCadena(Importe,' ','0')+'H'
               +Truncar(Concepto,26,' ')
FicheroTexto.Escribir(TextoFichero)
TotalReg=TotalReg+1;
```

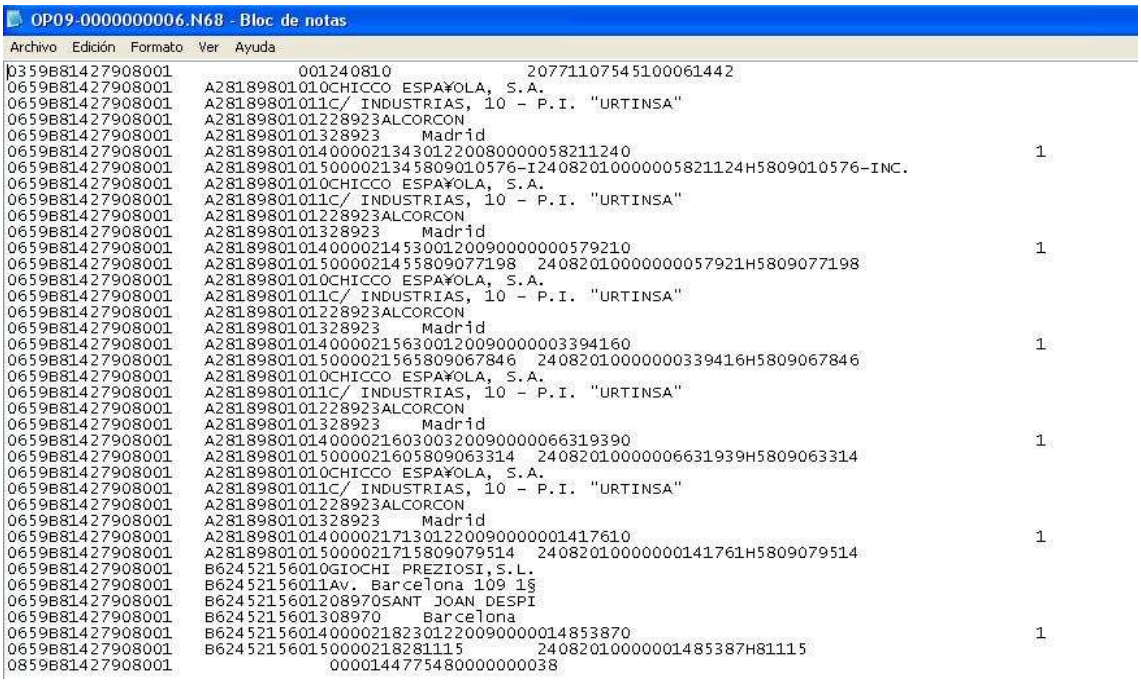
Código Original

```
GLSetup.GET;
IF CheckErrors THEN
  Relat := '1'
ELSE
  Relat := '0';

"Payment Order".FINDFIRST();
//IF NOT DocMisc.CheckCurrency("Payment Order"."Currency Code",LCY) THEN
//  ERROR (Text1100001);
IsEuro := DocMisc.GetRegisterCode("Payment Order"."Currency
Code",RegisterCode,RegisterString);
IF RegisterCode <> 0 THEN
  RegisterString := '56'
ELSE
  RegisterString := '06';


TESTFIELD("Bank Account No.");
BankAcc.GET("Bank Account No.");


CCCBankNo := BankAcc."CCC Bank No.";
CCCBankNo := PADSTR(",MAXSTRLEN(CCCBankNo) - STRLEN(CCCBankNo),'0') +
CCCBankNo;

CCCBankBranchNo := BankAcc."CCC Bank Branch No.";
CCCBankBranchNo := PADSTR(",MAXSTRLEN(CCCBankBranchNo) -
STRLEN(CCCBankBranchNo),'0') + CCCBankBranchNo;

CCCAccNo := BankAcc."CCC Bank Account No.";
CCCAccNo := PADSTR(",MAXSTRLEN(CCCAccNo) - STRLEN(CCCAccNo),'0') +
CCCAccNo;

CCCControlDigits := BankAcc."CCC Control Digits";
CCCControlDigits := PADSTR(",MAXSTRLEN(CCCControlDigits) -
STRLEN(CCCControlDigits),'0') + CCCControlDigits;
CCCIBAN := COPYSTR(BankAcc.IBAN,1,4);

IF "Posting Date" = 0D THEN
  PostDate := PADSTR(",6,'0')
ELSE
  PostDate := FORMAT("Posting Date",0,Text1100002);

OutText := '03' + '59' + PADSTR(VATRegNo,9,' ') + BankSufix + PADSTR(",12,' ') + '001' +
  FORMAT(DeliveryDate,0,Text1100002) + PADSTR(",9,' ') + PADSTR(CCCIBAN,4,' ') +
  CCCBankNo + CCCBankBranchNo + CCCControlDigits + CCCAccNo + PADSTR(",30,' ');
OutFile.WRITE(OutText);
TotalReg := TotalReg + 1;
```

```
ProvAnt:=';
VctoAnt:=0D;

// ILR. 31/08/09
// Marcamos la remesa cómo generrada.
"Fichero Generado":=TRUE;
MODIFY;

// Grabamos un comentario con la fecha y la hora de la creación del mismo.
nvLin:=0;
rcdComentario.RESET;
rcdComentario.SETRANGE("BG/PO No.","Payment Order"."No.");
rcdComentario.SETRANGE(Type,rcdComentario.Type::Payable);
IF rcdComentario.FINDLAST THEN
  nvLin:=rcdComentario."Line No.";

nvLin+=10000;

rcdComentario.INIT;
rcdComentario."BG/PO No.":="Payment Order"."No.";
rcdComentario.Type:=rcdComentario.Type::Payable;
rcdComentario."Line No.":=nvLin;
rcdComentario.Date:=TODAY;
rcdComentario.Comment:=FORMAT(CURRENTDATETIME)+ ' - ' + USERID + ' - ' +
'Genración de fichero ' + ExternalFile +'.';
rcdComentario.INSERT;

TESTFIELD("Payment Method Code");
//DocType2 := DocMisc.DocType2("Payment Method Code");

TESTFIELD("Account No.");
Vendor.GET("Account No.");

VATRegVend := Vendor."VAT Registration No.";
VATRegVend := PADSTR(',MAXSTRLEN(VATRegVend) - STRLEN(VATRegVend),' ') +
VATRegVend;
RmgAmount := PADSTR(',MAXSTRLEN(RmgAmount) - STRLEN(RmgAmount),'0') +
RmgAmount;

IF NOT Pais.GET(Vendor."Country Code") THEN BEGIN
  Pais.INIT;
  Pais.Code := ';
END;

CASE TRUE OF
  LCY = LCY::Peseta:
    IF IsEuro THEN BEGIN
      TotalAmount := TotalAmount + "Remaining Amount";
      ImportDoc := "Remaining Amount";
      RmgAmount := EuroAmount (ImportDoc);
    END ELSE BEGIN
      RmgAmount := FORMAT("Remaining Amt. (LCY)" * 100,12,Text1100000);
      TotalAmount := TotalAmount + "Remaining Amount";
```

```
    END;
  LCY = LCY::Euro:
    IF IsEuro THEN BEGIN
      TotalAmount := TotalAmount + "Remaining Amount";
      ImportDoc := "Remaining Amount";
      RmgAmount := EuroAmount (ImportDoc);
    END ELSE  BEGIN
      RmgAmount := FORMAT("Remaining Amount" * 100,12,Text1100000);
      TotalAmount := TotalAmount + "Remaining Amount";
    END;
  LCY = LCY::Other:
    IF IsEuro THEN  BEGIN
      TotalAmount := TotalAmount + "Remaining Amount";
      ImportDoc := "Remaining Amount";
      RmgAmount := EuroAmount (ImportDoc);
    END ELSE BEGIN
      RmgAmount := FORMAT("Remaining Amount" * 100,12,Text1100000);
      TotalAmount := TotalAmount + "Remaining Amount" * 100;
    END;
END;

OutText := '06' + '59' + PADSTR(VATRegNo,9,' ') + BankSufix + VATRegVend + '010' +
          PADSTR(Vendor.Name,40,' ') + PADSTR(',29,' ');
OutFile.WRITE(OutText);

TotalReg := TotalReg + 1;
TotalDocVend := TotalDocVend + 1;

OutText := '06' + '59' + PADSTR(VATRegNo,9,' ') + BankSufix + VATRegVend + '011' +
PADSTR(Vendor.Address,45,' ')
          + PADSTR(',24,' ');
OutFile.WRITE(OutText);
TotalReg := TotalReg + 1;

OutText := '06' + '59' + PADSTR(VATRegNo,9,' ') + BankSufix + VATRegVend + '012' +
PADSTR(Vendor."Post Code",5,' ') +
          PADSTR(Vendor.City,40,' ') + PADSTR(',24,' ');
OutFile.WRITE(OutText);
TotalReg := TotalReg + 1;

OutText := '06' + '59' + PADSTR(VATRegNo,9,' ') + BankSufix + VATRegVend + '013' +
PADSTR(Vendor."Post Code",9,' ')
          + PADSTR(Vendor.County,30,' ') + PADSTR(Pais.Name,20,' ') + PADSTR(',10,' ');
OutFile.WRITE(OutText);
TotalReg := TotalReg + 1;

    // Se calcula con el módulo a 7 de 9000 + el número de pagaré que queremos asignar a 7
dígitos
    // Lo  cogemos de la ficha del banco.
    BankAcc.TESTFIELD("Last Check No.");
    BankAcc."Last Check No.":=INCSTR(BankAcc."Last Check No.");
    BankAcc.MODIFY;
    NumPagareFichaBanco:=BankAcc."Last Check No.";
```

```
   // Rellenamos a 0 por la izquierda por si han modificado la ficha
   NumPagareFichaBanco:=PADSTR('',MAXSTRLEN(NumPagareFichaBanco)-
STRLEN(NumPagareFichaBanco),'0')+NumPagareFichaBanco;

   EVALUATE(Dividendo,'9000'+NumPagareFichaBanco );
   DigCtrlNumPago:= Dividendo MOD 7;
   NumPago:=NumPagareFichaBanco + FORMAT(DigCtrlNumPago);
   ProvAnt:=Vendor."No.";
   VctoAnt:="Cartera Doc."."Due Date";

END;

OutText := '06' + '59' + PADSTR(VATRegNo,9,' ') + BankSufix + VATRegVend + '014' +
NumPago +
   FORMAT("Due Date",0,Text1100005) + CONVERTSTR(RmgAmount,' ','0') + '0' +
PADSTR('',2,' ')
   + PADSTR('',6,' ') + PADSTR('',31,' ')+'1';
OutFile.WRITE(OutText);

TotalReg := TotalReg + 1;

{ MODIFICACIÓN - 22/03/04 }
Concepto := '';
NoDoc:='';
IF VendLedgEntry.GET("Entry No.") THEN
BEGIN
  Concepto := COPYSTR(VendLedgEntry."External Document No.",1,26);
  NoDoc := COPYSTR(VendLedgEntry."External Document No.",1,12);
END;

IF Concepto = '' THEN
  Concepto := COPYSTR(Description,1,26);
{ MODIFICACIÓN - 22/03/04 }

OutText := '06' + '59' + PADSTR(VATRegNo,9,' ') + BankSufix + VATRegVend + '015' +
NumPago +
        PADSTR(NoDoc,12,' ') + FORMAT(DeliveryDate,0,Text1100005) +
CONVERTSTR(RmgAmount,' ','0')
        + 'H' + PADSTR(Concepto,26,' ');

OutFile.WRITE(OutText);
TotalReg := TotalReg + 1;
```



OP09-0000000006.N68 - Bloc de notas

Archivo Edición Formato Ver Ayuda

```
0359B81427908001          001240810            20771107545100061442
0659B81427908001   A28189801010CHICCO ESPAÑOLA, S.A.
0659B81427908001   A28189801011C/ INDUSTRIAS, 10 - P.I. "URTINSA"
0659B81427908001   A2818980101228923ALCORCON
0659B81427908001   A2818980101328923    Madrid
0659B81427908001   A28189801014000021343012200800000058211240                                    1
0659B81427908001   A28189801015000021345809010576-I2408201000000058211124H5809010576-INC.
0659B81427908001   A28189801010CHICCO ESPAÑOLA, S.A.
0659B81427908001   A28189801011C/ INDUSTRIAS, 10 - P.I. "URTINSA"
0659B81427908001   A2818980101228923ALCORCON
0659B81427908001   A2818980101328923    Madrid
0659B81427908001   A28189801014000021453001200900000000579210                                    1
0659B81427908001   A28189801015000021455809077198  2408201000000000057921H5809077198
0659B81427908001   A28189801010CHICCO ESPAÑOLA, S.A.
0659B81427908001   A28189801011C/ INDUSTRIAS, 10 - P.I. "URTINSA"
0659B81427908001   A2818980101228923ALCORCON
0659B81427908001   A2818980101328923    Madrid
0659B81427908001   A28189801014000021563001200900000003394160                                    1
0659B81427908001   A28189801015000021565809067846  2408201000000000339416H5809067846
0659B81427908001   A28189801010CHICCO ESPAÑOLA, S.A.
0659B81427908001   A28189801011C/ INDUSTRIAS, 10 - P.I. "URTINSA"
0659B81427908001   A2818980101228923ALCORCON
0659B81427908001   A2818980101328923    Madrid
0659B81427908001   A28189801014000021603003200900000066319390                                    1
0659B81427908001   A28189801015000021605809063314  2408201000000006631939H5809063314
0659B81427908001   A28189801010CHICCO ESPAÑOLA, S.A.
0659B81427908001   A28189801011C/ INDUSTRIAS, 10 - P.I. "URTINSA"
0659B81427908001   A2818980101228923ALCORCON
0659B81427908001   A2818980101328923    Madrid
0659B81427908001   A28189801014000021713012200900000001417610                                    1
0659B81427908001   A28189801015000021715809079514  2408201000000000141761H5809079514
0659B81427908001   B62452156010GIOCHI PREZIOSI,S.L.
0659B81427908001   B62452156011Av. Barcelona 109 1§
0659B81427908001   B624521560120897OSANT JOAN DESPI
0659B81427908001   B6245215601308970    Barcelona
0659B81427908001   B62452156014000021823012200900000014853870                                    1
0659B81427908001   B62452156015000021828111S       2408201000000001485387H81115
0859B81427908001              00001447754800000000038
```

Ilustración 49 Ejemplo fichero generado ejecutando la norma 68

Norma34

PseudoCódigo

```
ConfigEmpresa.Coge
Si TieneErrores entonces
   Relat='1'
Sino
   Relat='0'
OrdenPago se posiciona en el primer registro
EsEuro=CodigoRegistro(OrdenPago.CodigoConcurrencia,CodigoRegistro,
                      CadenaRegistro)
Si CodigoRegistro<>0 entonces
    CadenaRegistro='56'
Sino
   CadenaRegistro='06'

Tablabanco.Coge(OrdenPago.NoCuentaBanco)
CuentaBanco=Tablabanco.NoBanco
CuentaBanco=TRUNCAR('',MaximaLongitud(CuentaBanco)-Longitud(CuentaBanco),
                    '0')+CuentaBanco
CuentaSucursal=Tablabanco.NoSucursal
CuentaSucursal=TRUNCAR('',MaximaLongitud(CuentaSucursal)-
                       Longitud(CuentaSucursal),'0')+CuentaSucursal
CuentaBanco=Tablabanco.NoCuentaBanco
NoCuentaBanco=TRUNCAR('',MaximaLongitud(NoCuentaBanco)-
                        Longitud(NoCuentaBanco), '0')+NoCuentaBanco
ControlDigitos=Tablabanco.ControlDigitos
ControlDigitos =TRUNCAR('',MaximaLongitud(ControlDigitos)-
                       Longitud(ControlDigitos),'0')+ ControlDigitos
IBAN=CopiarCadena(Tablabanco.IBAN,1,4)
Si DocCartera.FechaRegistro=0D entonces
     FechRegistro=TRUNCAR('',6,'0')
```

```
Sino
    FechRegistro=Formateo(OrdenPago.FechaRegistro,0, <Day,2><month,2><year>)
TextoFichero='03'+CadenaRegistro+NoRegIVA+Truncar('',12,' ')+'001'+
                Formateo(FechaEntrega,0, <Day,2><month,2><year>)+Fechregistro
                +CuentaBanco+CuentaSucursal+NoCuentaBanco+Relat+
                Truncar('',3,' ')+DigitoControl+Truncar('',7,' ')
FicheroTexto.Escribir(TextoFichero)
TotalReg=TotalReg+1
TextoFichero='03'+CadenaRegistro+NoRegIVA+Truncar('',12,' ')+'002'+
                Truncar(Empresa.Nombre,36,'')+Truncar('',7,' ')
FicheroTexto.Escribir(TextoFichero)
TotalReg=TotalReg+1
TextoFichero='03'+CadenaRegistro+NoRegIVA+Truncar('',12,' ')+'003'+
                Truncar(Empresa.Direccion,36,'')+Truncar('',7,' ')
FicheroTexto.Escribir(TextoFichero)
TotalReg=TotalReg+1
TextoFichero='03'+CadenaRegistro+NoRegIVA+Truncar('',12,' ')+'004'+
                Truncar(Empresa.CodigoPostal+' '+Empresa.Ciudad,36,'')
                +Truncar('',7,' ')
FicheroTexto.Escribir(TextoFichero)
TotalReg=TotalReg+1

OrdenPago.FicheroGenerado=Verdadero
Modificar registro OrdenPago
nvLin=0
TablaComentario se inicializa
TablaComentario se filtra No sea OrdenPago.No
TablaComentario se filtra Tipo sea Pago
Si encuentra ultimo registro entonces
    nvLin=TablaComentario.NoLinea
nvLin=nvLin+10000
TablaComentario se inicializa
TablaComentario.No=OrdenPago.No
TablaComentario.Tipo=Pago
TablaComentario.NoLinea=nvLin
TablaComentario.Fecha=Hoy
TablaComentario.Comentario=Formateo(Hoy)+' – '+Usuario+' – '+Generacion de
                                Fichero'+FicheroTexto+'.';
Insertar registro TablaComentario
TipoDoc=TipoDocumento(DocCartera.MetodoPago)
TablaProveedor.Coge(DocCartera.NoCuenta)
RegProvIVA=TablaProveedor.RegistroIVA
RegProvIVA=TRUNCAR('',Maxlongitud(RegProvIVA)-Longitud(RegProvIVA),' ')+
                RegProvIVA
Importe=TRUNCAR('',Maxlongitud(Importe)-Longitud(Importe),'0 ')+Importe
Caso Verdadero
   Moneda=Peseta: Si esEuro entonces
                     TotalImporte=TotalImporte+DocCartera.ImporteRestante
                     ImporteDocumento=ImporteRestante
                     Importe=EuroImporte(ImporteDocumento)
                  Sino
                     Importe=Formateo(ImporteRestante*100,12, <Integer>)
                     TotalImporte=TotalImporte+DocCartera.ImporteRestante
   Moneda=Euro:    Si esEuro entonces
```

```
                    TotalImporte=TotalImporte+DocCartera.ImporteRestante
                    ImporteDocumento=ImporteRestante
                    Importe=EuroImporte(ImporteDocumento)
                  Sino
                    Importe=Formateo(ImporteRestante*100,12, <Integer>)
                    TotalImporte=TotalImporte+DocCartera.ImporteRestante

Moneda=Otro:      Si esEuro entonces
                    TotalImporte=TotalImporte+DocCartera.ImporteRestante
                    ImporteDocumento=ImporteRestante
                    Importe=EuroImporte(ImporteDocumento)
                  Sino
                    Importe=Formateo(ImporteRestante*100,12, <Integer>)
                    TotalImporte=TotalImporte+DocCartera.ImporteRestante
Caso TipoDoc
  '3': NoCuentaProv='0000000010'
       ControlDigitosProv=''
       Prov2=''
       NoSucursalProv=''
       VctoPagare=Formateo(DocCartera.FechaVcto,0, <Day,2><month,2><year>)
  '4': CodigoBancoProv=DocCartera.CodigoBancoProv
       Si CuentaProveedor.Coge(DocCartera.NoCuenta,CodigoBancoProv)
          NoCuentaProv=CuentaProveedor.NoCuentaBanco
          ControlDigitosProv=CuentaProveedor.ControlDigitos
          Prov2=CuentaProveedor.NoCuentaBanco
          NoSucursalProv=CuentaProveedor.NoSucursal
          NoCuentaProv=Truncar('',Maxlongitud(NoCuentaProv)-
                       Longitud(NoCuentaProv),'0')+NoCuentaProv;
          NoCuentaProv=Truncar('',Maxlongitud(ControlDigitos)-
                       Longitud(ControlDigitos),'0')+ControlDigitos;
       Sino
          NoCuentaProv=''
          ControlDigitosProv=''
          Prov2=''
          NoSucursalProv=''
   '5': NoCuentaProv='0000000010'
        ControlDigitosProv=''
        Prov2=''
        NoSucursalProv=''
Si EsEuro entonces
  Caso TipoDoc
     '3':TipoDoc2='58'
     '4': TipoDoc2='57'
     '5': TipoDoc2='56'
Sino
  Si TipoDoc='4'
     TipoDoc2='07'
  Sino
     TipoDoc2='06'
Truncar(VctoPagare,7,'');
ZonaF1=ConvertirCadena(Importe,' ','0')
ZonaF2=ConvertirCadena(Truncar(Prov2,4,' '),' ','0')
ZonaF3= ConvertirCadena(Truncar(NoSucursalProv,4,' '),' ','0')
ZonaF4= ConvertirCadena(Truncar(NoCuentaProv,10,' '),' ','0')
```

```
ZonaF5=’1’
ZonaF6=’9’
ZonaF7=’+’
ZonaF8=Truncar(ControlDigitos,2,’ ‘)
ZonaG=Truncar(VctoPagare,7,’ ‘)
TextoFichero=’06’+TipoDoc2+NoRegIVA+RegProvIVA+’010’+ZonaF1+ZonaF2
                +ZonaF3+ZonaF4+ZonaF5+ZonaF6+ZonaF7+ZonaF8+ZonaG
FicheroTexto.Escribir(TextoFichero)
TotalReg=TotalReg+1
TotalDocProv=TotalDocProv+1
TextoFichero=’06’+TipoDoc2+NoRegIVA+RegProvIVA+’011’+
                Truncar(Proveedor.Nombre,36,’’)+Truncar(‘’,7,’ ‘)
FicheroTexto.Escribir(TextoFichero)
TotalReg=TotalReg+1
Si TipoDoc en[‘06’,’56’,’58’] entonces
    TextoFichero=’06’+TipoDoc2+NoRegIVA+RegProvIVA+’012’+
                Truncar(Proveedor.Direccion,36,’’)+Truncar(‘’,7,’ ‘)
    FicheroTexto.Escribir(TextoFichero)
    TotalReg=TotalReg+1
    Si Proveedor.Direccion2<>’’ entonces
       TextoFichero=’06’+TipoDoc2+NoRegIVA+RegProvIVA+’013’+
                    Truncar(Proveedor.Direccion2,36,’’)+Truncar(‘’,7,’ ‘)
       FicheroTexto.Escribir(TextoFichero)
       TotalReg=TotalReg+1
    TextoFichero=’06’+TipoDoc2+NoRegIVA+RegProvIVA+’014’+
                  Truncar(Proveedor.CodPostal+’ ‘+Proveedor.Ciudad,36,’’)
                  +Truncar(‘’,7,’ ‘)
    FicheroTexto.Escribir(TextoFichero)
    TotalReg=TotalReg+1
    Si Proveedor.Provincia<>’’ entonces
       TextoFichero=’06’+TipoDoc2+NoRegIVA+RegProvIVA+’015’+
                Truncar(Proveedor.Provincia,36,’’)+Truncar(‘’,7,’ ‘)
       FicheroTexto.Escribir(TextoFichero)
       TotalReg=TotalReg+1
    TablaMovProv se inicializa
    TablaMovProv se filtra NoDoc sea DocCartera.NoDocumento
    TablaMovProv se filtra NoProv sea DocCartera.NoCuenta
    Si encuentra registros TablaMovProv entonces
        cvDocExt=TablaMovProv.NoDocExt+’-‘+
        cvDocExt=cvDocExt+CopiarCadena(CarteraDoc.Descripcion,1,
                  MaxLongitud(cvDocExt)-longitud(cvDocExt))
       TextoFichero=’06’+TipoDoc2+NoRegIVA+RegProvIVA+’016’+
                    Truncar(cvDocExt,36,’ ‘)+Truncar(‘’,7,’ ‘)
       FicheroTexto.Escribir(TextoFichero)
       TotalReg=TotalReg+1
```

Código Original

```
GLSetup.GET;
IF CheckErrors THEN
  Relat := '1'
ELSE
  Relat := '0';
"Payment Order".FINDFIRST();
```

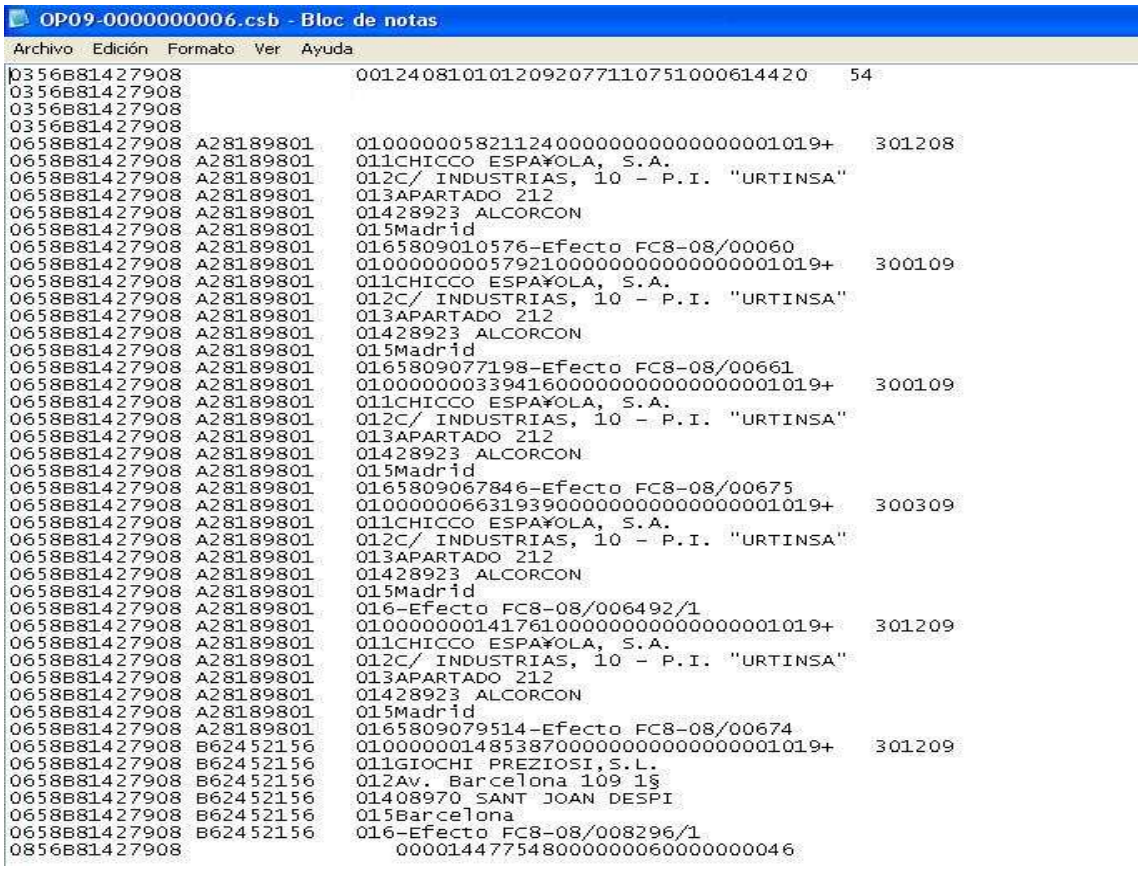
```
IsEuro := DocMisc.GetRegisterCode("Payment Order"."Currency
Code",RegisterCode,RegisterString);
IF RegisterCode <> 0 THEN
  RegisterString := '56'
ELSE
  RegisterString := '06';

TESTFIELD("Bank Account No.");
BankAcc.GET("Bank Account No.");

CCCBankNo := BankAcc."CCC Bank No.";
CCCBankNo := PADSTR('',MAXSTRLEN(CCCBankNo) - STRLEN(CCCBankNo),'0') +
CCCBankNo;

CCCBankBranchNo := BankAcc."CCC Bank Branch No.";
CCCBankBranchNo := PADSTR('',MAXSTRLEN(CCCBankBranchNo) -
STRLEN(CCCBankBranchNo),'0') + CCCBankBranchNo;

CCCAccNo := BankAcc."CCC Bank Account No.";
CCCAccNo := PADSTR('',MAXSTRLEN(CCCAccNo) - STRLEN(CCCAccNo),'0') +
CCCAccNo;

CCCControlDigits := BankAcc."CCC Control Digits";
CCCControlDigits := PADSTR('',MAXSTRLEN(CCCControlDigits) -
STRLEN(CCCControlDigits),'0') + CCCControlDigits;

IF "Posting Date" = 0D THEN
  PostDate := PADSTR('',6,'0')
ELSE
  PostDate := FORMAT("Posting Date",0,Text1100002);

OutText := '03' + RegisterString + VATRegNo + PADSTR('',12,' ') + '001' +
  FORMAT(DeliveryDate,0,Text1100002) + PostDate +
  CCCBankNo + CCCBankBranchNo + CCCAccNo + Relat + PADSTR('',3,' ') +
CCCControlDigits + PADSTR('',7,' ');
OutFile.WRITE(OutText);
TotalReg := TotalReg + 1;

OutText := '03' + RegisterString + VATRegNo + PADSTR('',12,' ') + '002' +
  PADSTR(CompanyInfo.Name,36,' ') + PADSTR('',7,' ');
OutFile.WRITE(OutText);
TotalReg := TotalReg + 1;

OutText := '03' + RegisterString + VATRegNo + PADSTR('',12,' ') + '003' +
  PADSTR(CompanyInfo.Address,36,' ') + PADSTR('',7,' ');
OutFile.WRITE(OutText);
TotalReg := TotalReg + 1;

OutText := '03' + RegisterString + VATRegNo + PADSTR('',12,' ') + '004' +
  PADSTR(CompanyInfo."Post Code" + ' ' + CompanyInfo.City,36,' ') + PADSTR('',7,' ');
OutFile.WRITE(OutText);
TotalReg := TotalReg + 1;

"Fichero Generado":=TRUE;
```

```
MODIFY;

nvLin:=0;
rcdComentario.RESET;
rcdComentario.SETRANGE("BG/PO No.","Payment Order"."No.");
rcdComentario.SETRANGE(Type,rcdComentario.Type::Payable);
IF rcdComentario.FINDLAST THEN
  nvLin:=rcdComentario."Line No.";
nvLin+=10000;
rcdComentario.INIT;
rcdComentario."BG/PO No.":="Payment Order"."No.";
rcdComentario.Type:=rcdComentario.Type::Payable;
rcdComentario."Line No.":=nvLin;
rcdComentario.Date:=TODAY;
rcdComentario.Comment:=FORMAT(CURRENTDATETIME)+ ' - ' + USERID + ' - ' +
'Genración de fichero ' + ExternalFile +'.';
rcdComentario.INSERT;

TESTFIELD("Payment Method Code");
DocType2 := DocMisc.DocType2("Payment Method Code");

TESTFIELD("Account No.");
Vendor.GET("Account No.");

VATRegVend := Vendor."VAT Registration No.";
VATRegVend := VATRegVend + PADSTR(',MAXSTRLEN(VATRegVend) -
STRLEN(VATRegVend));

RmgAmount := PADSTR(',MAXSTRLEN(RmgAmount) - STRLEN(RmgAmount),'0') +
RmgAmount;

CASE TRUE OF
 LCY = LCY::Peseta:
   IF IsEuro THEN BEGIN
     TotalAmount := TotalAmount + "Remaining Amount";
     RmgAmount := EuroAmount ("Remaining Amount");
   END ELSE BEGIN
     RmgAmount := FORMAT("Remaining Amt. (LCY)" * 100,12,Text1100000);
     TotalAmount := TotalAmount + "Remaining Amount";
   END;
 LCY = LCY::Euro:
   IF IsEuro THEN BEGIN
     TotalAmount := TotalAmount + "Remaining Amount";
     RmgAmount := EuroAmount ("Remaining Amount");
   END ELSE  BEGIN
     RmgAmount := FORMAT("Remaining Amount" * 100,12,Text1100000);
     TotalAmount := TotalAmount + "Remaining Amount";
   END;
 LCY = LCY::Other:
   IF IsEuro THEN  BEGIN
     TotalAmount := TotalAmount + "Remaining Amount";
     RmgAmount := EuroAmount ("Remaining Amount");
   END ELSE BEGIN
     RmgAmount := FORMAT("Remaining Amount" * 100,12,Text1100000);
```

```
    TotalAmount := TotalAmount + "Remaining Amount" * 100;
   END;
END;

VctoPagare:='';
CASE DocType2 OF
  '3': BEGIN
        VendCCCAccNo := '0000000010';
        VendCCCControlDigits := '';
        Vendor2 := '';
        VendCCCBankBranchNo := '';
        VctoPagare:=FORMAT("Due Date",0,Text1100002)
       END;
  '4': BEGIN
        VendBankAccCode := "Cust./Vendor Bank Acc. Code";
        IF VendBankAcc.GET("Account No.",VendBankAccCode) THEN BEGIN
          VendCCCAccNo := VendBankAcc."CCC Bank Account No.";
          VendCCCControlDigits := VendBankAcc."CCC Control Digits";
          Vendor2 := VendBankAcc."CCC Bank No.";
          VendCCCBankBranchNo := VendBankAcc."CCC Bank Branch No.";

          VendCCCAccNo          :=          PADSTR('',MAXSTRLEN(VendCCCAccNo)          -
STRLEN(VendCCCAccNo),'0') + VendCCCAccNo;
          VendCCCControlDigits:=PADSTR('',MAXSTRLEN(VendCCCControlDigits)          -
STRLEN(VendCCCControlDigits),'0') +VendCCCControlDigits;

        END ELSE BEGIN
          VendCCCAccNo := '';
          VendCCCControlDigits := '';
          Vendor2 := '';
          VendCCCBankBranchNo := '';
        END;
       END;

   '5': BEGIN
        VendCCCAccNo := '';
        VendCCCControlDigits := '';
        Vendor2 := '';
        VendCCCBankBranchNo := '';
       END;
END;

IF IsEuro THEN
BEGIN
  CASE DocType2 OF
    '3': DocType := '58';                    // Pagaré
    '4': DocType := '57';                    // Cheque Bancario
    '5': BEGIN
          DocType := '56';                   // Transferencia (validamos ccc proveedor)
          VendBankAcc.TESTFIELD("CCC Bank Account No.");
         END;
  END;
END
ELSE     // PESETAS.
```

```
BEGIN
  IF DocType2 = '4' THEN
    DocType := '07'
  ELSE BEGIN
    DocType := '06';
    VendBankAcc.TESTFIELD("CCC Bank Account No.");
  END;
END;

////////////////////////////////////////////////////////
// TIPO DE DATO = 010: Datos del abono
////////////////////////////////////////////////////////
ZonaF1:= CONVERTSTR(RmgAmount,' ','0');
ZonaF2:= CONVERTSTR(PADSTR(Vendor2,4,' '),' ','0');
ZonaF3:= CONVERTSTR(PADSTR(VendCCCBankBranchNo,4,' '),' ','0');
ZonaF4:= CONVERTSTR(PADSTR(VendCCCAccNo,10,' '),' ','0');
ZonaF5:= '1';
ZonaF6:= '9';
ZonaF7:= '+ ';
ZonaF8:= PADSTR(VendCCCControlDigits,2,' ') ;
ZonaG:=PADSTR(VctoPagare,7,' ');

OutText := '06' + DocType + VATRegNo + VATRegVend + '010'
  + ZonaF1
  + ZonaF2
  + ZonaF3
  + ZonaF4
  + ZonaF5
  + ZonaF6
  + ZonaF7
  + ZonaF8
  + ZonaG;

OutFile.WRITE(OutText);
TotalReg := TotalReg + 1;
TotalDocVend := TotalDocVend + 1;

////////////////////////////////////////////////////////
// TIPO DE DATO = 011: Nombre del proveedor
////////////////////////////////////////////////////////
OutText := '06' + DocType + VATRegNo + VATRegVend + '011' +
PADSTR(Vendor.Name,36,' ') + PADSTR('',7,' ');
OutFile.WRITE(OutText);
TotalReg := TotalReg + 1;

IF DocType IN ['06','56','58'] THEN
BEGIN
  ////////////////////////////////////////////////////////
  // TIPO DE DATO = 012: Dirección del proveedor
  ////////////////////////////////////////////////////////
  OutText := '06' + DocType + VATRegNo + VATRegVend + '012' +
PADSTR(Vendor.Address,36,' ') + PADSTR('',7,' ');
  OutFile.WRITE(OutText);
  TotalReg := TotalReg + 1;
```

```
  ////////////////////////////////////////////////////////
  // TIPO DE DATO = 013: Dirección2 del proveedor
  ////////////////////////////////////////////////////////
  IF Vendor."Address 2" <> '' THEN BEGIN
    OutText := '06' + DocType + VATRegNo + VATRegVend + '013' +
PADSTR(Vendor."Address 2",36,' ') + PADSTR('',7,' ');
    OutFile.WRITE(OutText);
    TotalReg := TotalReg + 1;
  END;

  ////////////////////////////////////////////////////////
  // TIPO DE DATO = 014: Codigo postal y calle
  ////////////////////////////////////////////////////////
  OutText := '06' + DocType + VATRegNo + VATRegVend + '014' + PADSTR(Vendor."Post
Code" + ' ' + Vendor.City,36,' ')
    + PADSTR('',7,' ');
  OutFile.WRITE(OutText);
  TotalReg := TotalReg + 1;

  ////////////////////////////////////////////////////////
  // TIPO DE DATO = 015: Provincia
  ////////////////////////////////////////////////////////
  IF Vendor.County <> '' THEN
  BEGIN
    OutText := '06' + DocType + VATRegNo + VATRegVend + '015' +
PADSTR(Vendor.County,36,' ')
      + PADSTR('',7,' ');
    OutFile.WRITE(OutText);
    TotalReg := TotalReg + 1;
  END;
  rcdmovProv.RESET;
  rcdmovProv.SETCURRENTKEY("Document No.","Document Type","Vendor No.");
  rcdmovProv.SETRANGE("Document No.","Document No.");
  rcdmovProv.SETRANGE("Vendor No.","Account No.");
  IF rcdmovProv.FINDFIRST THEN
  BEGIN
    cvDocExt:= rcdmovProv."External Document No." + '-';
    cvDocExt:= cvDocExt + COPYSTR("Cartera
Doc.".Description,1,MAXSTRLEN(cvDocExt) - STRLEN(cvDocExt)) ;
    OutText := '06' + DocType + VATRegNo + VATRegVend + '016' +
PADSTR(cvDocExt,36,' ') + PADSTR('',7,' ');
    OutFile.WRITE(OutText);
    TotalReg := TotalReg + 1;
  END;
END;
```



OP09-0000000006.csb - Bloc de notas

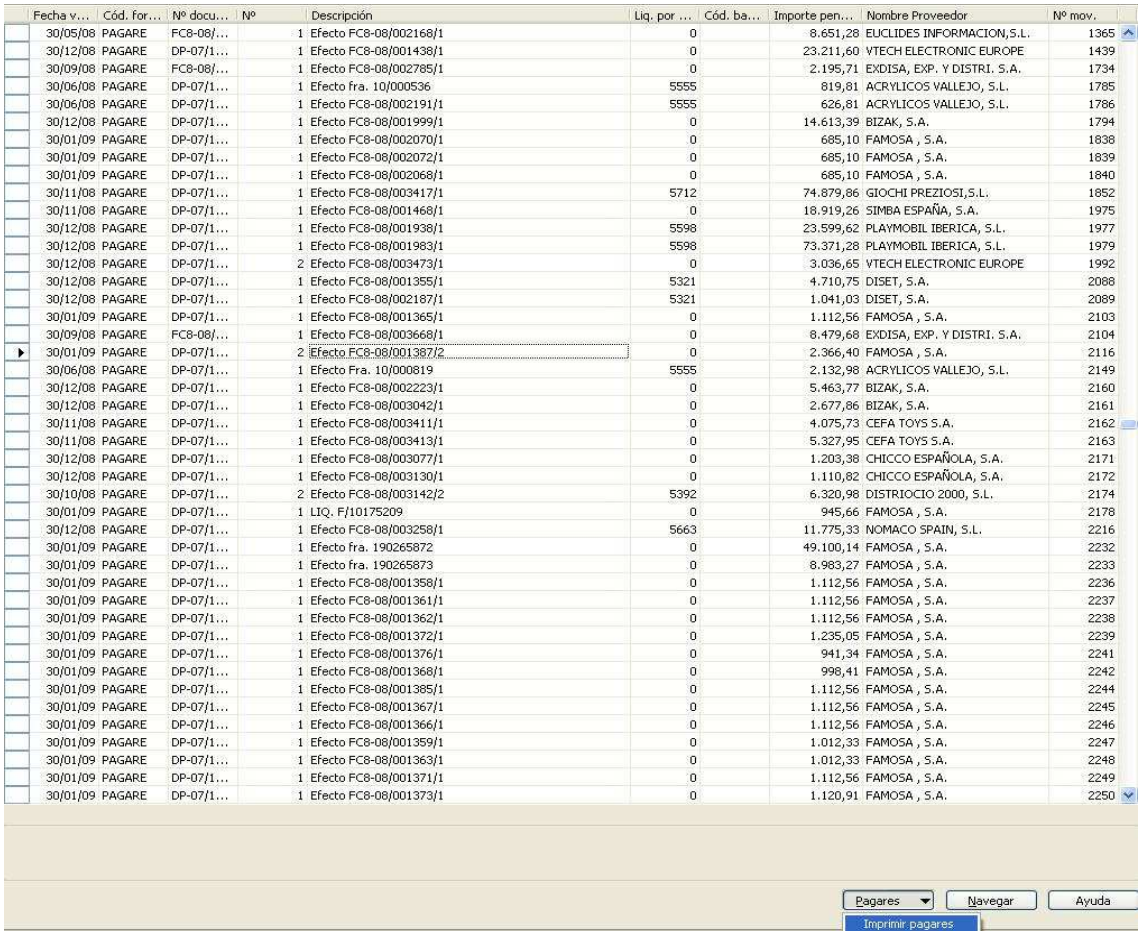
Archivo Edición Formato Ver Ayuda

```
0356B81427908              0012408101012092077110751000614420   54
0356B81427908
0356B81427908
0356B81427908
0658B81427908 A28189801    0100000058211240000000000000000001019+   301208
0658B81427908 A28189801    011CHICCO ESPA¥OLA, S.A.
0658B81427908 A28189801    012C/ INDUSTRIAS, 10 - P.I. "URTINSA"
0658B81427908 A28189801    013APARTADO 212
0658B81427908 A28189801    01428923 ALCORCON
0658B81427908 A28189801    015Madrid
0658B81427908 A28189801    0165809010576-Efecto FC8-08/00060
0658B81427908 A28189801    0100000000579210000000000000000001019+   300109
0658B81427908 A28189801    011CHICCO ESPA¥OLA, S.A.
0658B81427908 A28189801    012C/ INDUSTRIAS, 10 - P.I. "URTINSA"
0658B81427908 A28189801    013APARTADO 212
0658B81427908 A28189801    01428923 ALCORCON
0658B81427908 A28189801    015Madrid
0658B81427908 A28189801    0165809077198-Efecto FC8-08/00661
0658B81427908 A28189801    0100000003394160000000000000000001019+   300109
0658B81427908 A28189801    011CHICCO ESPA¥OLA, S.A.
0658B81427908 A28189801    012C/ INDUSTRIAS, 10 - P.I. "URTINSA"
0658B81427908 A28189801    013APARTADO 212
0658B81427908 A28189801    01428923 ALCORCON
0658B81427908 A28189801    015Madrid
0658B81427908 A28189801    0165809067846-Efecto FC8-08/00675
0658B81427908 A28189801    0100000066319390000000000000000001019+   300309
0658B81427908 A28189801    011CHICCO ESPA¥OLA, S.A.
0658B81427908 A28189801    012C/ INDUSTRIAS, 10 - P.I. "URTINSA"
0658B81427908 A28189801    013APARTADO 212
0658B81427908 A28189801    01428923 ALCORCON
0658B81427908 A28189801    015Madrid
0658B81427908 A28189801    016-Efecto FC8-08/006492/1
0658B81427908 A28189801    0100000001417610000000000000000001019+   301209
0658B81427908 A28189801    011CHICCO ESPA¥OLA, S.A.
0658B81427908 A28189801    012C/ INDUSTRIAS, 10 - P.I. "URTINSA"
0658B81427908 A28189801    013APARTADO 212
0658B81427908 A28189801    01428923 ALCORCON
0658B81427908 A28189801    015Madrid
0658B81427908 A28189801    0165809079514-Efecto FC8-08/00674
0658B81427908 B62452156    0100000014853870000000000000000001019+   301209
0658B81427908 B62452156    011GIOCHI PREZIOSI,S.L.
0658B81427908 B62452156    012Av. Barcelona 109 1§
0658B81427908 B62452156    01408970 SANT JOAN DESPI
0658B81427908 B62452156    015Barcelona
0658B81427908 B62452156    016-Efecto FC8-08/008296/1
0856B81427908                 000014477548000000060000000046
```

Ilustración 50 Ejemplo de fichero generado ejecutando la norma34

Pagaré

En los pedidos de compra se configura la fecha de vencimiento del pago, porque en el caso de que el proveedor tenga configurada como forma de pago PAGARÉ, una vez se cumpla esa fecha de vencimiento, se debe entregar un pagaré para paliar la deuda con el proveedor. En este caso, existe la opción de agrupar los pagos pendientes con el proveedor e imprimir un pagaré únicamente o un pagaré por cada pago pendiente. Para ello, existe un formulario donde se indica cada uno de los pagos de los que se quiere emitir el pagaré. Una vez señalados los movimientos, existe la opción Imprimir Pagaré dentro del formulario. Dentro de esa función se deben configurar una serie de opciones, como son: Banco con el que se va a realizar la gestión, si se quiere agrupar movimiento e imprimir pagaré por agrupación o imprimir un pagaré por cada movimiento y si se quiere que salga un texto al final del pagaré. Una vez configuradas las opciones, se deben imprimir los pagarés. Entonces cuando se inicia la impresión, el sistema inserta los movimientos de pagarés que van asociados, para que quede constancia que se han emitido y poder consultar el saldo del proveedor.



Fecha v...	Cód. for...	Nº docu...	Nº	Descripción	Liq. por ...	Cód. ba...	Importe pen...	Nombre Proveedor	Nº mov.
30/05/08	PAGARE	FC8-08/...	1	Efecto FC8-08/002168/1	0		8.651,28	EUCLIDES INFORMACION,S.L.	1365
30/12/08	PAGARE	DP-07/1...	1	Efecto FC8-08/001438/1	0		23.211,60	VTECH ELECTRONIC EUROPE	1439
30/09/08	PAGARE	FC8-08/...	1	Efecto FC8-08/002785/1	0		2.195,71	EXDISA, EXP. Y DISTRI. S.A.	1734
30/06/08	PAGARE	DP-07/1...	1	Efecto fra. 10/000536	5555		819,81	ACRYLICOS VALLEJO, S.L.	1785
30/06/08	PAGARE	DP-07/1...	1	Efecto FC8-08/002191/1	5555		626,81	ACRYLICOS VALLEJO, S.L.	1786
30/12/08	PAGARE	DP-07/1...	1	Efecto FC8-08/001999/1	0		14.613,39	BIZAK, S.A.	1794
30/01/09	PAGARE	DP-07/1...	1	Efecto FC8-08/002070/1	0		685,10	FAMOSA , S.A.	1838
30/01/09	PAGARE	DP-07/1...	1	Efecto FC8-08/002072/1	0		685,10	FAMOSA , S.A.	1839
30/01/09	PAGARE	DP-07/1...	1	Efecto FC8-08/002068/1	0		685,10	FAMOSA , S.A.	1840
30/11/08	PAGARE	DP-07/1...	1	Efecto FC8-08/003417/1	5712		74.879,86	GIOCHI PREZIOSI,S.L.	1852
30/11/08	PAGARE	DP-07/1...	1	Efecto FC8-08/001468/1	0		18.919,26	SIMBA ESPAÑA, S.A.	1975
30/12/08	PAGARE	DP-07/1...	1	Efecto FC8-08/001938/1	5598		23.599,62	PLAYMOBIL IBERICA, S.L.	1977
30/12/08	PAGARE	DP-07/1...	1	Efecto FC8-08/001983/1	5598		73.371,28	PLAYMOBIL IBERICA, S.L.	1979
30/12/08	PAGARE	DP-07/1...	2	Efecto FC8-08/003473/1	0		3.036,65	VTECH ELECTRONIC EUROPE	1992
30/12/08	PAGARE	DP-07/1...	1	Efecto FC8-08/001355/1	5321		4.710,75	DISET, S.A.	2088
30/12/08	PAGARE	DP-07/1...	1	Efecto FC8-08/002187/1	5321		1.041,03	DISET, S.A.	2089
30/01/09	PAGARE	DP-07/1...	1	Efecto FC8-08/001365/1	0		1.112,56	FAMOSA , S.A.	2103
30/09/08	PAGARE	FC8-08/...	1	Efecto FC8-08/003668/1	0		8.479,68	EXDISA, EXP. Y DISTRI. S.A.	2104
30/01/09	PAGARE	DP-07/1...	2	Efecto FC8-08/001387/2	0		2.366,40	FAMOSA , S.A.	2116
30/06/08	PAGARE	DP-07/1...	1	Efecto Fra. 10/000819	5555		2.132,98	ACRYLICOS VALLEJO, S.L.	2149
30/12/08	PAGARE	DP-07/1...	1	Efecto FC8-08/002223/1	0		5.463,77	BIZAK, S.A.	2160
30/12/08	PAGARE	DP-07/1...	1	Efecto FC8-08/003042/1	0		2.677,86	BIZAK, S.A.	2161
30/11/08	PAGARE	DP-07/1...	1	Efecto FC8-08/003411/1	0		4.075,73	CEFA TOYS S.A.	2162
30/11/08	PAGARE	DP-07/1...	1	Efecto FC8-08/003413/1	0		5.327,95	CEFA TOYS S.A.	2163
30/12/08	PAGARE	DP-07/1...	1	Efecto FC8-08/003077/1	0		1.203,38	CHICCO ESPAÑOLA, S.A.	2171
30/12/08	PAGARE	DP-07/1...	1	Efecto FC8-08/003130/1	0		1.110,82	CHICCO ESPAÑOLA, S.A.	2172
30/10/08	PAGARE	DP-07/1...	2	Efecto FC8-08/003142/2	5392		6.320,98	DISTRIOCIO 2000, S.L.	2174
30/01/09	PAGARE	DP-07/1...	1	LIQ. F/10175209	0		945,66	FAMOSA , S.A.	2178
30/12/08	PAGARE	DP-07/1...	1	Efecto FC8-08/003258/1	5663		11.775,33	NOMACO SPAIN, S.L.	2216
30/01/09	PAGARE	DP-07/1...	1	Efecto fra. 190265872	0		49.100,14	FAMOSA , S.A.	2232
30/01/09	PAGARE	DP-07/1...	1	Efecto fra. 190265873	0		8.983,27	FAMOSA , S.A.	2233
30/01/09	PAGARE	DP-07/1...	1	Efecto FC8-08/001358/1	0		1.112,56	FAMOSA , S.A.	2236
30/01/09	PAGARE	DP-07/1...	1	Efecto FC8-08/001361/1	0		1.112,56	FAMOSA , S.A.	2237
30/01/09	PAGARE	DP-07/1...	1	Efecto FC8-08/001362/1	0		1.112,56	FAMOSA , S.A.	2238
30/01/09	PAGARE	DP-07/1...	1	Efecto FC8-08/001372/1	0		1.235,05	FAMOSA , S.A.	2239
30/01/09	PAGARE	DP-07/1...	1	Efecto FC8-08/001376/1	0		941,34	FAMOSA , S.A.	2241
30/01/09	PAGARE	DP-07/1...	1	Efecto FC8-08/001368/1	0		998,41	FAMOSA , S.A.	2242
30/01/09	PAGARE	DP-07/1...	1	Efecto FC8-08/001385/1	0		1.112,56	FAMOSA , S.A.	2244
30/01/09	PAGARE	DP-07/1...	1	Efecto FC8-08/001367/1	0		1.112,56	FAMOSA , S.A.	2245
30/01/09	PAGARE	DP-07/1...	1	Efecto FC8-08/001366/1	0		1.112,56	FAMOSA , S.A.	2246
30/01/09	PAGARE	DP-07/1...	1	Efecto FC8-08/001359/1	0		1.012,33	FAMOSA , S.A.	2247
30/01/09	PAGARE	DP-07/1...	1	Efecto FC8-08/001363/1	0		1.012,33	FAMOSA , S.A.	2248
30/01/09	PAGARE	DP-07/1...	1	Efecto FC8-08/001371/1	0		1.112,56	FAMOSA , S.A.	2249
30/01/09	PAGARE	DP-07/1...	1	Efecto FC8-08/001373/1	0		1.120,91	FAMOSA , S.A.	2250

Pagares | Navegar | Ayuda
Imprimir pagares

Ilustración 51 Formulario dónde se debe señalar los movimientos que se quiere imprimir pagaré

Ilustración 52 Configurar opciones para la impresión de pagaré

PseudoCódigo

```
Si Tablabanco.No='' entonces
    ERROR('Debes seleccionar un banco')
Si NoPagare='' entonces
    ERROR('No hay pagaré seleccionado')
Si optnumMovs=0 entonces
   ERROR('No hay movimientos imprimibles')
Si optImporte<=0 entonces
   ERROR('El importe no debe ser 0')
ConfigEmpresa se posiciona primer registro
MovPagare se inicializa
Si posiciona ultimo registro MovPagare entonces
    Primero=MovPagare.NoMov+1
Sino
    Primero=1
Num=Primero
Si DocCartera.Liqporpagare=0 entonces
   MovPagare se inicializa
   MovPagare.NoMov=num
    MovPagare.NoBanco=Tablabanco.No
    AuxBanco.Coge(Tablabanco.No)
    AuxBanco.UltNumPagare=Incrementar(AuxBanco.UltNumPagare)
    Modificar registro AuxBanco
    MovPagare.NoCheque=NoPagare
    MovPagare.FechaRegistro=HOY
    MovPagare.TipoDocumento=Efecto
    MovPagare.NoDocumento=DocCartera.NoDocumento
    MovPagare.Descripcion=DocCartera.Descripcion
    MovPagare.NoEfecto=DocCartera.No
    MovPagare.EstadodelCheque=Impreso
    MovPagare.FechaCheque=HOY
    MovPagare.Pendiente=Verdadero
    MovPagare.CodProv=DocCartera.NoCuenta
    MovPagare.Liq por NoDoc=DocCartera.Liq por NoDoc
    Insertar registro MovPagare
    Si No Vista Preliminar entonces
       NoPagare=CogerSiguiente(optserie,OD,TRUE)
       optDigito='8200'+NoPagare
       optDigito=optDigito MOD 7
       NoPagare=NoPagare+'.'+optDigito
    Sino
       AuxNoPagare=Incrementar(AuxNoPagare)
       optDigito='8200'+AuxNoPagare
       optDigito=optDigito MOD 7
       AuxNoPagare=AuxNoPagare+'.'+optDigito
    TablaEfecto.Copy(DocCartera)
    Si UnoporProv entonces
        MovPagaré.Descripcion='AGRUPACION'
       TablaEfecto se filtra NoCuenta sea DocCartera.NoCuenta
       TablaEfecto se filtra FechaVcto sea DocCartera.FechaVcto
       TablaEfecto se filtra Liq por Pagare sea 0
       Si encuentra registros TablaEfecto entonces
          I=0
          Repetir
```

```
                TablaEfecto.Liq por Pagare=MovPagare.NoMov
                TablaEfecto.Impreso=Verdadero
                Modifica registro TablaEfecto
                MovPagare.Importe=MovPagare.Importe+TablaEfecto.ImportePendiente
                MovPagare.Importe(Base)=MovPagare.Importe(Base)+
                                        TablaEfecto.ImportePendiente(Base)
            Hasta (registro TablaEfecto sea vacío) O (I>=15)
        MovPagare.FechaVcto=DocCartera.FechaVcto
        Modificar registro MovPagare
    Sino
        DocCartera.Liq por Pagare=MovPagare.NoMov
        DocCartera.Impreso=Verdadero
        Modificar registro DocCartera
        MovPagare.NoEfecto=DocCartera.No
        MovPagare.Importe=DocCartera.ImportePendiente
        MovPagare.Importe(Base)=DocCartera.ImportePendiente(Base)
        MovPagare.FechaVcto=DocCartera.FechaVcto
        Modificar registro MovPagare
    Ultimo=num
    MovPagare se inicializa
    Si encuentra ultimo registro MovPagare entonces
        Num=MovPagare.NoMov+1
    Sino
        Num=1
Texto1='Muy señores nuestros:'+'\'+'          Adjuntamos pagaré por importe de '+
        Formateo(MovPagare.Importe)+' Euros. que cancela las facturas que a
        continuación detallamos.'
Texto2='Esperando sea de su conformidad, les saludamos atentamente.';
TextoCheque=MovPagare.NoCheque
Si DocCartera.CodigoConcurrencia='' entonces
    ConvertirTexto(DescripLinea,MovPagare.Importe,1,80)
Sino
    ConvertirTexto(DescripLinea,MovPagare.Importe,2,80)
DescripLinea[1]=Mayusculas(DescripLinea[1])
DescripLinea[2]=Mayusculas(DescripLinea[2])
TextoImporteCheque=Formateo(Redóndear(MovPagare.Importe,0.01,'='))
Si POS(TextImporteCheque,',')=0 entonces
   TextoImporteCheque=TextoImporteCheque+',00'
Sino Si POS(TextoImporteCheque,',')=Longitud(TextoImporteCheque)-1 entonces
   TextoImporteCheque=TextoImporteCheque+'0'
Longtext=Longitud(TextoImporteCheque)
For i=1 to 13-longtext hacer
     TextoImporteCheque='*'+TextoImporteCheque
TextoFechaCheque=Formateo(MovPagare.FechaCheque,0,<Day> de <Month Text> de
                    <Year4>.')
Tablabanco2.Coge(MovPagare.NoBanco)
PoblacionProvincia=Tablabanco2.CodigoPostal+'-'+Tablabanco2.Ciudad+ '('+
                    Tablabanco2.Provincia+')';
OfiControl=Tablabanco2.NoSucursal+'-'+Tablabanco2.DigitosControl
TextoImporteCheque='//'+TextoImporteCheque+'//'
txtColetilla=''
Si Coletilla entonces
   txtColetilla='NO A LA ORDEN'
Si FacturaCompra.Coge(TablaEfecto.NoDocumento) entonces
```

```
   NumFactProv=FacturaCompra.NoFactura+'/'+TablaEfecto.No
Sino
   NumFactProv=''
Si NumFactProv='' entonces
   MovProv.Coge(Tablaefecto.NoDetalle)
   NumFactProv=MovProv.NoDocExterno
```

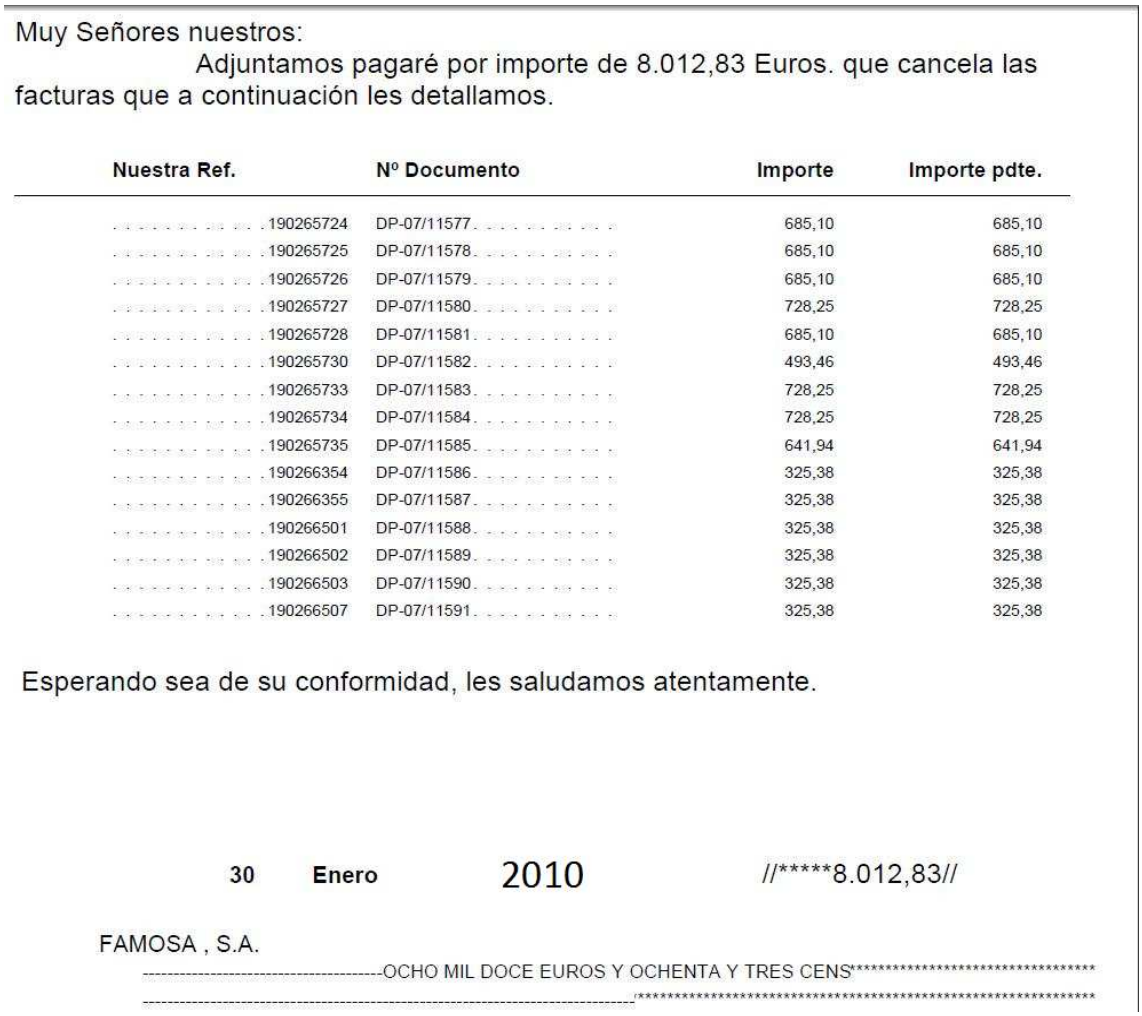
Código Original

```
// Validaciones del formulario
IF CtaBanco2."No." = '' THEN
  ERROR('Error. Debes seleccionar un banco');

IF NoPagare  ='' THEN
  ERROR('Error. No hay número de pagaré seleccionado');

IF optNumMovs = 0 THEN
  ERROR('Error. No hay movimientos imprimibles en la selección');

IF optImporte <= 0 THEN
  ERROR('Error. El importe del pagaré no puede ser 0');

MovPagare.RESET;
IF MovPagare.FINDLAST THEN
 Primero := MovPagare."Nº mov." + 1
ELSE
 Primero := 1;
num := Primero;

CLEAR(cduSerie);
IF CurrReport.PREVIEW THEN
  cvAuxPagare:=cduSerie.GetNextNo(optSerie,0D,FALSE);


IF "Cartera Doc."."Liq. por pagare" = 0 THEN
 BEGIN
   MovPagare.INIT;
   MovPagare."Nº mov." := num;
   MovPagare."Nº banco" := CtaBanco2."No.";
   CtaBanco.GET(CtaBanco2."No.");
   CtaBanco."Ult. numero pagaré" := INCSTR(CtaBanco."Ult. numero pagaré");
   CtaBanco.MODIFY;
   MovPagare."Nº cheque" := NoPagare;
   MovPagare."Fecha registro" := WORKDATE;
   MovPagare."Tipo documento" := MovPagare."Tipo documento"::Efecto;
   MovPagare."Nº documento" := "Cartera Doc."."Document No.";
   MovPagare.Descripción := "Cartera Doc.".Description;
   MovPagare."Nº efecto" := "Cartera Doc."."No.";
   MovPagare."Estado del cheque" := MovPagare."Estado del cheque"::Impreso;
   MovPagare."Fecha cheque" :=  WORKDATE;
   MovPagare.Pendiente := TRUE;
   MovPagare."Cod. Proveedor" := "Cartera Doc."."Account No.";
   MovPagare."Liq. por Nº documento" := "Cartera Doc."."Liq. por Nº documento";
   MovPagare.INSERT;
```

```
  IF NOT CurrReport.PREVIEW THEN
  BEGIN
    NoPagare:=cduSerie.GetNextNo(optSerie,0D,TRUE);
    optDigito:=DigitoControlPagare(NoPagare);
    NoPagare:=NoPagare + '.' + optDigito;
  END
  ELSE
  BEGIN
    cvAuxPagare:=INCSTR(cvAuxPagare);
    optDigito:=DigitoControlPagare(cvAuxPagare);
    NoPagare:=cvAuxPagare + '.' + optDigito;
  END;


  Efecto2.COPY("Cartera Doc.");
  IF Unoporprov THEN
   BEGIN
    MovPagare.Descripción := 'AGRUPACIÓN';
    Efecto2.SETRANGE("Account No.","Cartera Doc."."Account No.");
   Efecto2.SETRANGE("Due Date","Cartera Doc."."Due Date");
    Efecto2.SETRANGE("Liq. por pagare" ,0);
    IF Efecto2.FINDSET() THEN
    BEGIN
     I:=0;
     REPEAT
       Efecto2."Liq. por pagare" := MovPagare."Nº mov.";
       Efecto2.Impreso := TRUE;
       Efecto2.MODIFY;
       MovPagare.Importe := MovPagare.Importe + Efecto2."Remaining Amount";
       MovPagare."Importe (DL)" := MovPagare."Importe (DL)" + Efecto2."Remaining Amt.
(LCY)";
       I+=1;
     UNTIL (Efecto2.NEXT = 0) OR (I>=15);
    END;
    MovPagare."Fecha de vencimiento" := "Cartera Doc."."Due Date";
    MovPagare.MODIFY;
  END
  ELSE
    BEGIN
      "Cartera Doc."."Liq. por pagare" := MovPagare."Nº mov.";
      "Cartera Doc.".Impreso := TRUE;
      "Cartera Doc.".MODIFY;
      MovPagare."Nº efecto" := "Cartera Doc."."No.";
      MovPagare.Importe := "Cartera Doc."."Remaining Amount";
      MovPagare."Importe (DL)" := "Cartera Doc."."Remaining Amt. (LCY)";
      MovPagare."Fecha de vencimiento" := "Cartera Doc."."Due Date";
      MovPagare.MODIFY;
    END;
  Ultimo := num;
  MovPagare.RESET;
  IF MovPagare.FINDLAST THEN
    num := MovPagare."Nº mov." + 1
  ELSE
    num := 1;
```

```
 END;


VarTexto1 := 'Muy Señores nuestros:'+ '\' +
          '                Adjuntamos pagaré por importe de ' +
          FORMAT("Mov. pagares".Importe) +' Euros. que cancela las facturas que a
continuación les detallamos.';

VarTexto2 := 'Esperando sea de su conformidad, les saludamos atentamente.';
TexNoCheque := "Mov. pagares"."Nº cheque";

CLEAR(ConvertirAtexto);
IF "Cartera Doc."."Currency Code" = '' THEN
  ConvertirAtexto.NaT(DescriLinea,"Mov. pagares".Importe,1,80)
ELSE
  ConvertirAtexto.NaT(DescriLinea,"Mov. pagares".Importe,2,80);
DescriLinea[1] := UPPERCASE(DescriLinea[1]);
DescriLinea[2] := UPPERCASE(DescriLinea[2]);


TexImporteCheque := FORMAT(ROUND("Mov. pagares".Importe,0.01,'='));
IF STRPOS(TexImporteCheque,',') = 0 THEN
  TexImporteCheque := TexImporteCheque + ',00'
ELSE  IF STRPOS(TexImporteCheque,',') =  STRLEN(TexImporteCheque)-1 THEN
    TexImporteCheque := TexImporteCheque + '0';
longtext := STRLEN(TexImporteCheque);
FOR i := 1 TO 13 - longtext DO
   TexImporteCheque := '*' + TexImporteCheque;

TextFechaCheque := FORMAT("Mov. pagares"."Fecha cheque",0,'<Day> de <Month Text> de
<Year4>.');

CLEAR(FormatDirec);
FormatDirec.Company(DirEmpresa,InfoEmpresa);
CtaBanco2.GET("Mov. pagares"."Nº banco");

CLEAR(FormatDirec);
Prov.GET("Mov. pagares"."Cod. Proveedor");
Prov.TESTFIELD(Blocked,Prov.Blocked::'' '');
Prov.Contact := '';
FormatDirec.Vendor(DirCheque,Prov);
PoblaciónProvincia := CtaBanco2."Post Code" + '-' + CtaBanco2.City + ' (' +
CtaBanco2.County + ')';
OfiControl := CtaBanco2."CCC Bank Branch No." + '-' + CtaBanco2."CCC Control Digits";

longtext := STRLEN(TexImporteCheque);
FOR i := 1 TO 13 - longtext DO
   TexImporteCheque := '*' + TexImporteCheque;
TexImporteCheque := '//' + TexImporteCheque + '//';


txtColetilla:='';
IF bColetiila THEN
   txtColetilla:='NO A LA ORDEN';
```

```
IF RecFactCompra.GET(EfectoImp."Document No.") THEN
  NumFactProv := RecFactCompra."Vendor Invoice No." + '/' + EfectoImp."No."
ELSE
  NumFactProv := '';
IF NumFactProv = '' THEN
BEGIN
 MovProv.GET(EfectoImp."Entry No.");
 NumFactProv := MovProv."External Document No.";
END;
```



Muy Señores nuestros:

Adjuntamos pagaré por importe de 8.012,83 Euros. que cancela las facturas que a continuación les detallamos.

Nuestra Ref.	Nº Documento	Importe	Importe pdte.
190265724	DP-07/11577	685,10	685,10
190265725	DP-07/11578	685,10	685,10
190265726	DP-07/11579	685,10	685,10
190265727	DP-07/11580	728,25	728,25
190265728	DP-07/11581	685,10	685,10
190265730	DP-07/11582	493,46	493,46
190265733	DP-07/11583	728,25	728,25
190265734	DP-07/11584	728,25	728,25
190265735	DP-07/11585	641,94	641,94
190266354	DP-07/11586	325,38	325,38
190266355	DP-07/11587	325,38	325,38
190266501	DP-07/11588	325,38	325,38
190266502	DP-07/11589	325,38	325,38
190266503	DP-07/11590	325,38	325,38
190266507	DP-07/11591	325,38	325,38

Esperando sea de su conformidad, les saludamos atentamente.

30 Enero 2010 //*****8.012,83//

FAMOSA , S.A.

----------OCHO MIL DOCE EUROS Y OCHENTA Y TRES CENS**********

----------**********

Ilustración 53 Ejemplo de Pagaré que se genera al ejecutar la función de impresión

Presupuestos Compras

Los responsables de contabilidad al inicio del año deben configurar el presupuesto anual que va destinado a las compras. En primer lugar, deben definir cual será el importe total para el presupuesto de todo el año. En el momento en que se inserta esa cantidad, el sistema por defecto divide ese importe entre 12. En caso de que se quiera personalizar cada mes, como se muestra en la ilustración 54, donde se debe modificar el importe presupuestado que calculó el sistema, a partir del presupuesto anual.Una vez

configurado el presupuesto de cada mes, dentro de cada uno de ellos se debe configurar cuanto dinero irá destinado para cada una de las familias que se muestran en la ilustración 54 (Hobby, Juguetes, VideoJuegos, Disfraces,etc).

En esa pantalla, existen también otras funcionalidades: sacar un informe de lo que se lleva gastado de presupuesto por cada proveedor, actualizar consumos al momento y saber en tiempo real lo que se lleva gastado en cada mes y volver a reasignar el presupuesto de un mes, debido a que se han sobrepasado las previsiones y se debe destinar más dinero a una familia.

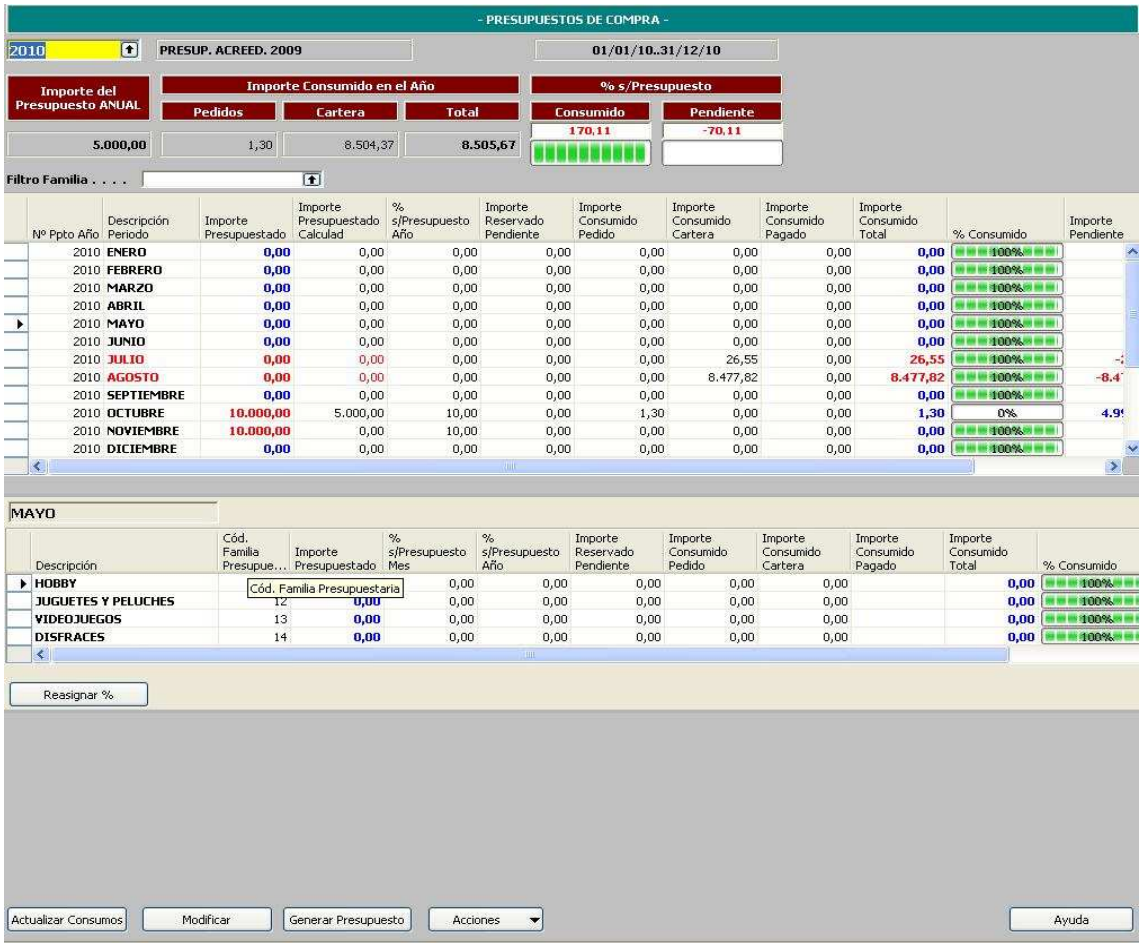


Ilustración 54 Configuración del Presupuesto para compras del año 2010

Actualizar Consumos(TablaPresupuesto)

PseudoCódigo

```
Si Confirma('¿Deseas actualizar los datos del presupuesto %1?',Falso,
                Formateo(TablaPresupuesto.NoPptoAño)) entonces
   CalcularDatos(TablaPresupuesto)
   TablaPresupuesto se inicializa
   TablaPresupuesto se filtra Activo sea Verdadero
   TablaPresupuesto se filtra NoPptoAño sea optAño
   TablaPresupuesto se calcula ImportePresupuestado, ImporteConsumidoPedido
```

ImporteConsumidoCartera, ImporteConsumidoTotal

Código Original

```
IF CONFIRM('¿Deseas actualizar los datos del presupuesto %1?',FALSE,FORMAT(rcdPresup."Nº Ppto Año")) THEN
BEGIN
  FunPPTO.CalcularDatos(rcdPresup);
  rcdPresup.RESET;
  rcdPresup.SETRANGE(Activo,TRUE);
  rcdPresup.SETRANGE(rcdPresup."Nº Ppto Año",optAño);
  IF rcdPresup.FINDFIRST THEN
    rcdPresup.CALCFIELDS("Importe Presupuestado","Importe Consumido Pedido","Importe Consumido Cartera","Importe Consumido Total");
END;
```

CalcularDatos(TablaPresupuesto)

PseudoCódigo

```
ConfiguracionCompras se posiciona primer registro
ConfiguracionCompras.SegundoControlPptoActivo=Falso
Modificar registro ConfiguraciónCompras
TablaPresup se inicializa
TablaPresup se filtra Activo sea Verdadero
TablaPresup se filtra NoPptoAño sea TablaPresupuesto.NoPptoAño
Si no encuentra registro TablaPresup entonces
    ERROR('No existe presupuesto activo')
FechaInicio=TablaPresup.FechaInicial
FechaFin=TablaPresup.FechaFinal
Si encuentra ultimo registro TablaPresup entonces
    FechaFin=TablaPresup.FechaFinal
TablaPresup se posiciona en el primer registro
TablaDetalle se bloquea tabla
TablaDetalle se inicializa
Repetir
   TablaDetalle se filtra NoPptoAño sea TablaPresup.NoPptoAño
   TablaDetalle se filtra Tipo sea <>Pagado Y <>Reservado
   Si encuentra registros TablaDetalle entonces
       Repetir
          BorraDetalle(TablaDetalle.Tipo,TablaDetalle.NoDocumento)
       Hasta registro TablaDetalle sea vacío
Hasta registro TablaPresup sea vacío
TablaEmpresa se inicializa
Si encuentra registros TablaEmpresa entonces
   Repetir
      TablaPedidos se inicializa
      TablaPedidos.CambiaEmpresa(TablaEmpresa.Nombre)
      TablaPedidos se filtra TipoDocumento sea Pedido
      TablaPedidos se filtra No sea PC*
      TablaPedidos se filtra FechaVcto sea >=FechaIni Y <=FechaFin
      TablaPedidos se filtra Subfamilia sea <>''
      TablaPedidos se filtra Validado sea Verdadero
      Si encuentra registros TablaPedidos entonces
          Repetir
             ImputarPlazos(TablaPedidos,Falso)
```

```
        Hasta registro TablaPedidos sea vacío
      TablaCartera se inicializa
      TablaCartera.CambioEmpresa(TablaEmpresa.Nombre)
      TablaCartera se filtra Tipo sea A pagar
      TablaCartera se filtra FechaVcto sea >=FechaIni Y <=FechaFin
      TablaCartera se filtra Subfamilia se <>''
      Si encuentra registros TablaCartera entonces
        Repetir
          GrabaDetalle(TablaCartera.CodigoSubfamilia,TablaCartera.FechaVcto,
                       TablaCartera.ImportePendiente,Cartera,TablaCartera.NoDoc,
                       TablaCartera.No, TablaCartera.NoCuenta,0D,
                       TablaCartera.CodigoDepartamento,Falso,
                       TablaCartera.Subfamilia)
          SumaImporte(TablaCartera.CodigoSubfamilia,TablaCartera.FechaVcto,
                       TablaCartera.Importe,Cartera)
        Hasta registro TablaCartera sea vacio
    Hasta registro TablaEmpresa sea vacío
    ConfiguracionCompras se posiciona primer registro
    ConfiguracionCompras.SegundoControlPptoActivo=Verdadero
    Modificar registro ConfiguraciónCompras
```

Código Original

```
PurchSetup.GET;
PurchSetup."Segundo Control Ppto. Activo":=FALSE;
PurchSetup.MODIFY;

// Vaciamos la tabla detalle (excepto la cartera) :ILR 17/03/09 ... Y Reservados.
rcdPresup.RESET;
rcdPresup.SETRANGE(Activo,TRUE);
rcdPresup.SETRANGE(rcdPresup."Nº Ppto Año",PARrcdPresup."Nº Ppto Año");
IF NOT rcdPresup.FINDFIRST() THEN
  ERROR('No existe ningún presupuesto activo');

dvInicioGlobal:=rcdPresup."Fecha Inicial";
dvFinGlobal:=rcdPresup."fecha final";

IF rcdPresup.FINDLAST() THEN
  dvFinGlobal:=rcdPresup."fecha final";
rcdPresup.FINDSET();

rcdDetalle.LOCKTABLE;
rcdDetalle.RESET;
REPEAT
 rcdDetalle.SETRANGE("Nº Ppto Año",rcdPresup."Nº Ppto Año");
 rcdDetalle.SETFILTER(Tipo,'<>%1                                                    &
<>%2',rcdDetalle.Tipo::Pagado,rcdDetalle.Tipo::Reservado);
 IF rcdDetalle.FINDSET THEN
 BEGIN
   REPEAT
     // En vez de hacer un delete, llamo a la función BorraDetalle para que controle los pedidos
programados.
     BorraDetalle(rcdDetalle.Tipo,rcdDetalle."Nº Documento");
   UNTIL rcdDetalle.NEXT=0;
 END;
```

```
UNTIL rcdPresup.NEXT=0;

//numLinea:=1;
rcdEmpresa.RESET;
IF rcdEmpresa.FINDSET() THEN
REPEAT
  rcdPedidos.RESET;
  rcdPedidos.CHANGECOMPANY(rcdEmpresa.Name);
  rcdPedidos.SETCURRENTKEY("Document Type","No.");
  rcdPedidos.SETRANGE("Document Type",rcdPedidos."Document Type"::Order);
  rcdPedidos.SETFILTER("No.", 'PC*');
  rcdPedidos.SETRANGE("Due Date",dvInicioGlobal,dvFinGlobal);
  rcdPedidos.SETFILTER(rcdPedidos."Shortcut Dimension 1 Code",'<> %1','');
  rcdPedidos.SETRANGE(Validado,TRUE);

  IF rcdPedidos.FINDSET() THEN
  REPEAT
    Ven.UPDATE(2,rcdPedidos."No.");
    ImputarPlazos(rcdPedidos,FALSE);
  UNTIL rcdPedidos.NEXT = 0;

  //Obtenemos el importe de los documento de cartera filtrados por "Fecha Vencimiento"
  rcdCartera.RESET;
  rcdCartera.CHANGECOMPANY(rcdEmpresa.Name);
  rcdCartera.SETCURRENTKEY(Type, "Bill Gr./Pmt. Order No.", "Collection Agent", "Due
Date", "Global Dimension 1 Code",
        "Global Dimension 2 Code", "Category Code", "Posting Date", "Document No.",
Accepted, "Currency Code", "Document Type");

  rcdCartera.SETFILTER(Type, 'A pagar');
  rcdCartera.SETFILTER("Bill Gr./Pmt. Order No.", '');
  rcdCartera.SETRANGE("Due Date", dvInicioGlobal,dvFinGlobal);
  rcdCartera.SETFILTER(rcdCartera."Global Dimension 1 Code",'<> %1','');

  IF rcdCartera.FINDSET() THEN
  REPEAT
    GrabaDetalle(DevDimValorPresup(rcdCartera."Global Dimension 1 Code"),
          rcdCartera."Due Date",rcdCartera."Remaining Amount",
          optTipo::Cartera,rcdCartera."Document No.",rcdCartera."No.",
          rcdCartera."Account No.",0D,rcdCartera."Global Dimension 2
Code",FALSE,rcdCartera."Global Dimension 1 Code");
    SumaImporte(DevDimValorPresup(rcdCartera."Global Dimension 1
Code"),rcdCartera."Due Date",rcdCartera."Remaining Amount",
          optTipo::Cartera);
  UNTIL (rcdCartera.NEXT=0);
UNTIL (rcdEmpresa.NEXT=0);
PurchSetup.GET;
PurchSetup."Segundo Control Ppto. Activo":=TRUE;
PurchSetup.MODIFY;
```

BorraDetalle(TablaDetalle.Tipo,TablaDetalle.NoDocumento)

PseudoCódigo

```
PresupDetalle se inicializa
PresupDetalle se filtra Tipo sea TablaDetalle.Tipo
PresupDetalle se filtra NoDocumento sea TablaDetalle.NoDocumento
Si encuentra registros PresupDetalle entonces
   Repetir
      Borramos Registro PresupDetalle
      Si Tipo=Pedido Y PedidoProgramado entonces
         ActualizaPptoProgramado(PresupDetalle.CodProv,PresupDetalle.NoPptoAño,
                                 PresupDetalle.NoPptoMes,
                                 PresupDetalle.CodFamilia,-(PresupDetalle.Importe)
   Hasta registro PresupDetalle sea vacío
```

Código Original

```
PPTODetalle.SETCURRENTKEY(PPTODetalle.Tipo,PPTODetalle."Nº Documento");
PPTODetalle.SETRANGE(PPTODetalle.Tipo,PARTipo);
PPTODetalle.SETFILTER(PPTODetalle."Nº Documento",PARNDoc);
IF PPTODetalle.FINDSET THEN
REPEAT
 PPTODetalle.DELETE();
 // Si es un pedido 'programado', hay que 'AMPLIAR' el presupuesto disponible para ese
proveedor-familia-mes
 // Uso la misma función pero la llamo en Negativo
 IF (PARTipo=PARTipo::Pedido) AND (PedidoProgramado(PARNDoc)) THEN
   ActualizaPptoProgramado(PPTODetalle."Cód. Proveedor",PPTODetalle."Nº Ppto Año",
                           PPTODetalle."Nº Ppto Mes",PPTODetalle."Cód. Familia
Presupuestaria",-PPTODetalle.Importe);
UNTIL PPTODetalle.NEXT=0;
```

GenerarPresupuesto

PseudoCódigo

```
Evaluar(FechaIni,'0101'+Formateo(TablaPresupuesto.Año)
FechaFin=Calcular('1A-1D,FechaIni)
Si No Confirma('¿Desear generar un presupuesto cuyo periodo sea desde %1 a%2?',
                Falso,FechaIni,FechaFin)
   ERROR('Cancelado')
TablaDetalle se inicializa
TablaDetalle se filtra NoPptoAño sea TablaPresupuesto.NoPptoAño
Si encuentra registros TablaDetalle entonces
   Borrar todos los registros TablaDetalle
TablaPresupuesto.FechaInicial=FechaIni
TablaPresupuesto.FechaFinal=FechaFin
Modificar registro TablaPresupuesto
nvImporteAnual=TablaPresupuesto.ImportePresupuestado
TablaCalendario se inicializa
TablaCalendario se filtra Tipo sea Mes
TablaCalendario se filtra PeriodoInicial sea >=FechaIni
Si encuentra registros TablaCalendario entonces
   Repetir
      i=i+1
      Si PresupMes.Coge(TablaPresupuesto.NoPptoAño,i) entonces
```

```
            Borrar registro PresupMes
        PresupMes se inicializa
        PresupMes.NoPptoAño=TablaPresupuesto.NoPptoAño
        PresupMes.NoPptoMes=i
        PresupMes.DescripcionPeriodo=Mayusculas(TablaCalendario.NombrePeriodo)
        PresupMes.FechaInicial=TablaCalendario.PeriodoInicio
        PresupMes.FechaFinal=TablaCalendario.PeriodoFinal
        PresupMes.Año=TablaPresupuesto.Año
        PresupMes.TipoPresupuesto=TablaPresupuesto.TipoPresupuesto
        Si nvImporteAnual<>0 entonces
           PresupMes.ImportePrespuestado=nvImporteAnual/12
           PresupMes.% s/Presupuesto Año=PresupMes.ImportePresupuestado*200/
                                                 nvImporteAnual
        Insertar registro PresupMes
        DimValFamilia se filtra CodigoDim sea SUBFAMILIA
        DimValFamilia se filtra ValorPresup sea Verdadero
        DimValFamilia se filtra TipoPresupuesto sea TablaPresupuesto.TipoPresupuesto
        nvnumFam=DimValFamilia.Contar
        Si encuentra registros DimValFamilia entonces
           Repetir
              Si DetalleFam.Coge(TablaPresupuesto.NoPptoAño,PresupMes.NoPptoMes,
                                    DimValFamilia.Codigo) entonces
                 Borrar registro DetalleFam
              DetalleFam.NoPptoAño=TablaPresupuesto.NoPptoAño
              DetalleFam.NoPptoMes=PresupMes.NoPptoMes
              DetalleFam.Año=TablaPresupuesto.Año
              DetalleFam.TipoPresupuesto=TablaPresupuesto.TipoPresupuesto
              DetalleFam.CodFamilia=DimValFamilia.Code
              Si nvImporteAnual<>0 entonces
                 DetalleFam.ImportePresupuestado=nvImporteAnual/12/nvNumFam
                 DetalleFam.% s/PresupuestoMes=100/nvNumFam
                 DetalleFam.% s/PresupuestoAño=100/12/nvNumFam
              Insertar registro DetalleFam
           Hasta registro DimValFamilia sea vacío
     Hasta registro TablaCalendario sea vacío O i=12
Si encuentra registros PresupMes entonces
     Repetir
        Recalcula()
        Modificar registro PresupMes
     Hasta registro PresupMes sea vacío
```

Código Original

```
EVALUATE(dvIni,'01/' + FORMAT(optIni +1) + '/' +  FORMAT(rcdPresup.Año));
dvFin:=CALCDATE('1A-1D',dvIni);
IF NOT CONFIRM('¿Deseas generar un presupuesto cuyo periodo sea desde %1 a
%2?',FALSE,dvIni,dvFin) THEN
  ERROR('cancelado');
// Vaciamos la tabla detalle
rcdDetalle.RESET;
rcdDetalle.SETRANGE("Nº Ppto Año",rcdPresup."Nº Ppto Año");
IF rcdDetalle.FIND('-') THEN
  rcdDetalle.DELETEALL();

rcdPresup."Fecha Inicial":=dvIni;
```

```
rcdPresup."fecha final":=dvFin;
rcdPresup.MODIFY;

nvImporteAnual:=rcdPresup."Importe Presupuestado";

rcdFecha.RESET;
rcdFecha.SETRANGE("Period Type",rcdFecha."Period Type"::Mes);
rcdFecha.SETFILTER("Period Start",'%1..',dvIni);
i:= 0;
IF rcdFecha.FIND('-') THEN
REPEAT
   i:= i+1;
   IF rcdMes.GET(rcdPresup."Nº Ppto Año",i) THEN
     rcdMes.DELETE;

   rcdMes.INIT;
   rcdMes."Nº Ppto Año":=rcdPresup."Nº Ppto Año";
   rcdMes."Nº Ppto Mes":=i;
   rcdMes."Descripción Periodo":=UPPERCASE(rcdFecha."Period Name");
   rcdMes."Fecha Inicial":=rcdFecha."Period Start";
   rcdMes."fecha final":=rcdFecha."Period End";
   rcdMes.Año:=rcdPresup.Año;
   rcdMes."Tipo Presupuesto":=rcdPresup."Tipo Presupuesto";
   IF nvImporteAnual <> 0 THEN
   BEGIN
    rcdMes."Importe Presupuestado":=nvImporteAnual / 12;
    rcdMes."% s/Presupuesto Año":=rcdMes."Importe Presupuestado" * 100 / nvImporteAnual;
   END;
   rcdMes.INSERT;

   DimValFam.SETRANGE(DimValFam."Dimension Code",'SUBFAMILIA');
   DimValFam.SETRANGE(DimValFam."Valor Presupuestario",TRUE);
   DimValFam.SETRANGE(DimValFam."Tipo Presupuesto",rcdPresup."Tipo Presupuesto");
   nvNumFam:=DimValFam.COUNT();
   IF DimValFam.FIND('-') THEN
   REPEAT
      IF rcdDetFam.GET(rcdPresup."Nº Ppto Año",rcdMes."Nº Ppto Mes",DimValFam.Code)
THEN
        rcdDetFam.DELETE;
      rcdDetFam."Nº Ppto Año":=rcdPresup."Nº Ppto Año";
      rcdDetFam."Nº Ppto Mes":=rcdMes."Nº Ppto Mes";
      rcdDetFam.Año:=rcdPresup.Año;
      rcdDetFam."Tipo Presupuesto":=rcdPresup."Tipo Presupuesto";
      rcdDetFam.VALIDATE("Cód. Familia Presupuestaria",DimValFam.Code);
      IF nvImporteAnual <> 0 THEN
      BEGIN
        rcdDetFam."Importe Presupuestado":=nvImporteAnual / 12 / nvNumFam;
        rcdDetFam."% s/Presupuesto Mes":=100 / nvNumFam;
        rcdDetFam."% s/Presupuesto Año":=100 / 12 / nvNumFam;
      END;
      rcdDetFam.INSERT;
   UNTIL DimValFam.NEXT=0;

UNTIL ((rcdFecha.NEXT =0) OR (i = 12));
```

```
IF rcdMes.FIND('-') THEN
REPEAT
   rcdMes.Recalcula();
   rcdMes.MODIFY;
UNTIL rcdMes.NEXT=0;
```

Recalcula

PseudoCódigo

```
Si TablaMes.Coge(PresupMes.NoPptoAño) entonces
    TablaMes calcula ImportePresupuestado
    PresupMes calcula ImporteConsumidoPedido,ImporteConsumidoCartera,
    ImporteConsumidoTotal
    Si TablaMes.ImportePresupuestado<>0 entonces
        PresupMes.% s/PresupuestoAño=PresupMes.ImportePresupuestado*100/
                                          TablaMes.ImportePresupuestado
        PresupMes.% desviacionAño=PresupMes.ImporteConsumidoTotal*100/
                                       TablaMes.ImportePresupuestado
```

Código Original

```
IF rcdCab.GET("Nº Ppto Año") THEN
BEGIN
 rcdCab.CALCFIELDS("Importe Presupuestado");
 CALCFIELDS("Importe Consumido Pedido","Importe Consumido Cartera","Importe
Consumido Total");
 IF rcdCab."Importe Presupuestado" <> 0 THEN
 BEGIN
   "% s/Presupuesto Año":=  "Importe Presupuestado" * 100 /rcdCab."Importe Presupuestado";
   "% desviacion Año":=          "Importe Consumido Total" * 100 /rcdCab."Importe
Presupuestado";
 END;
END;
```

Resumen Por Proveedor

En este apartado, se debe configurar una serie de opciones antes de ejecutar el informe. Se debe decidir si se quieren sacar todos los proveedores o filtrar una serie de ellos. Lo mismo ocurre con la subfamilia, en caso de que se quiera sacar el resumen de presupuesto de una subfamilia en concreto. Por último, debe poner desde que fecha se quiere sacar el informe.

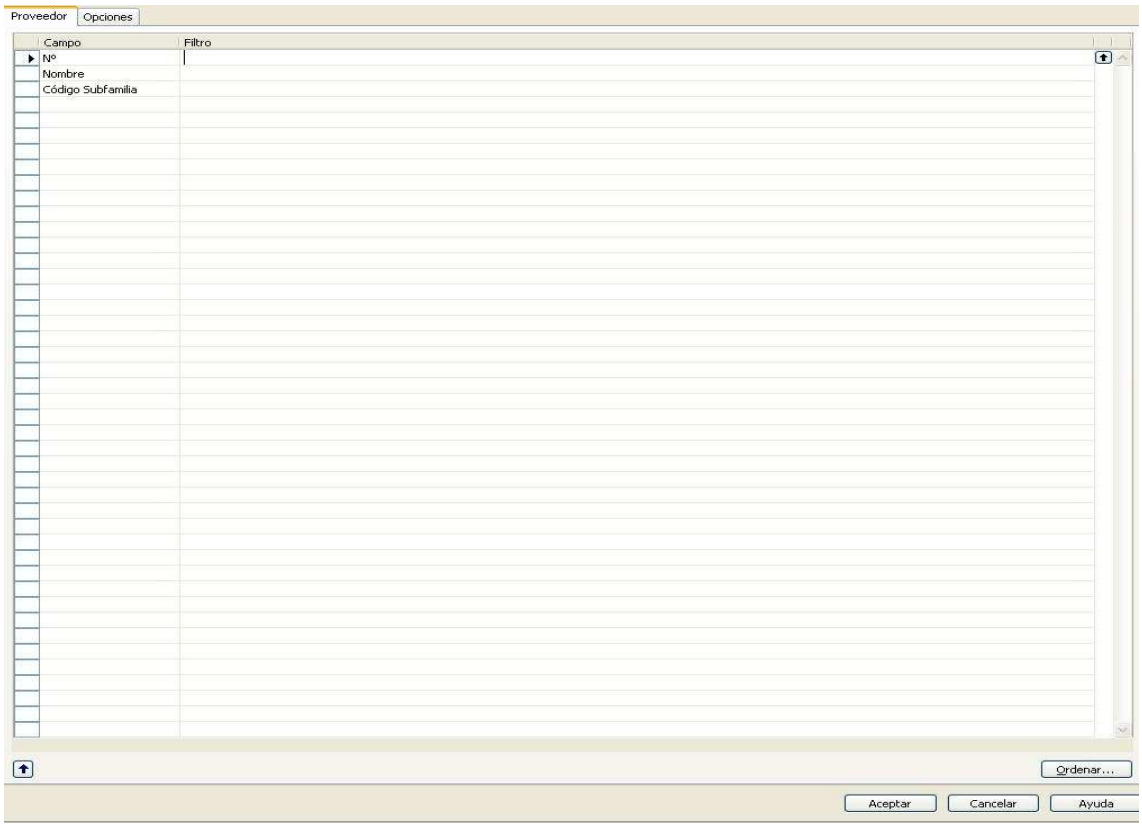


Ilustración 55 Pantalla donde se configuran las opciones del informe

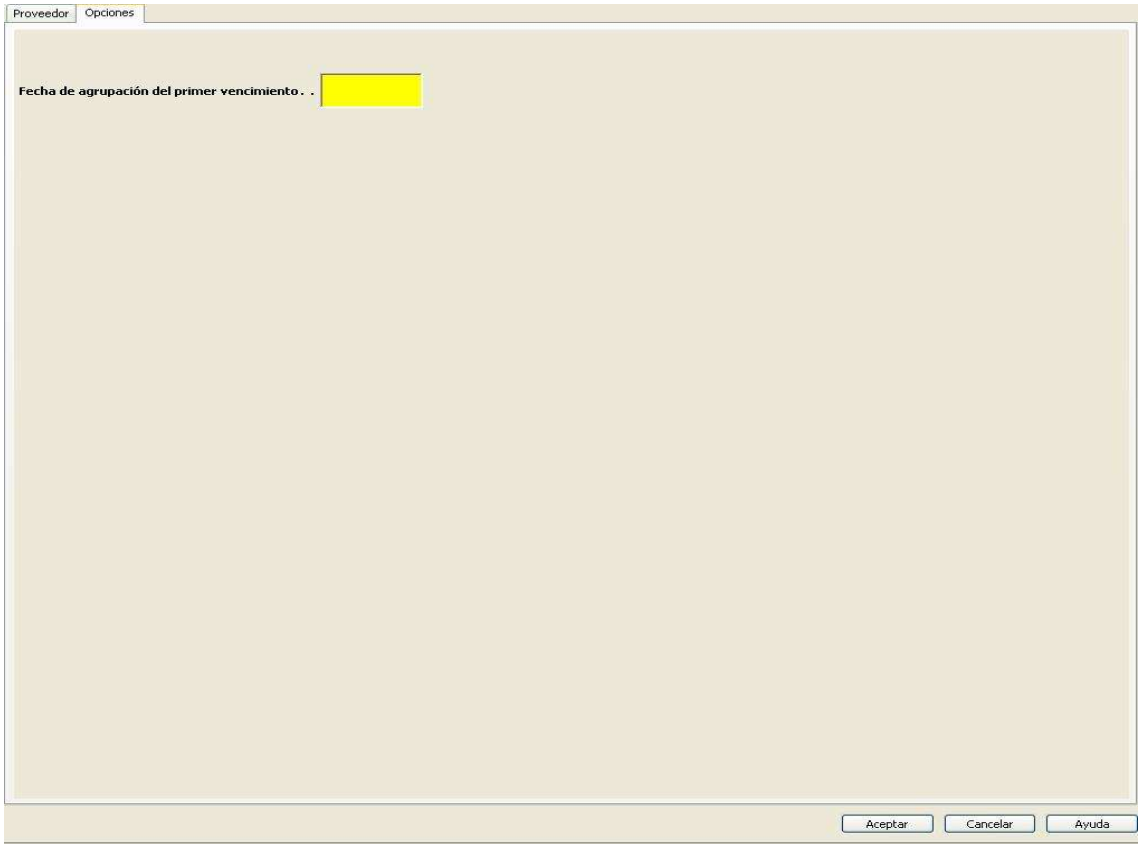


Ilustración 56 Pantalla donde se indica desde qué fecha se quiere sacar el informe

PseudoCódigo

```
Si optFechaAgrup=0D entonces
   ERROR('Debes seleccionar la fecha de agrupación')
Si No Crea(AplicacionExcel,Verdadero) entonces
    ERROR('Error al acceder a Excel')
AplicacionExcel.Visible(Verdadero)
LibroExcel=AplicacionExcel.Libro.Añadir;
HojaExcel=LibroExcel.Hoja.Numero
HojaExcel.Nombre='Pres.Compra'
Excell_Pinta(2,1,'Cod.',Verdadero,Verdadero)
Excell_Pinta(2,2,'Proveedor',Verdadero,Verdadero)
PresupuestoMes se inicializa
PresupuestoMes se filtra TipoPresupuesto sea Proveedores
PresupuestoMes se filtra FechaInicial sea mayor que optFechAgrup
i=3
Excell_Pinta(1,i,'Vencido a 31/12/2009',Verdadero,Verdadero)
HojaExcel.Rango('C1:F1').Combinar
HojaExcel.Rango('C1:F1').Alinear
Excell_Pinta(2,i,'EMITIDO',Verdadero,Verdadero)
Excell_Pinta(2,i+1,'PEDIDO',Verdadero,Verdadero)
Excell_Pinta(2,i+2,'CARTERA',Verdadero,Verdadero)
Excell_Pinta(2,i+3,'TOTAL',Verdadero,Verdadero)
i=i+4
Si encuentra registros PresupuestoMes entonces
    Repetir
       Si i<>2 entonces
          Excell_Pinta(1,i,PresupuestoMes.DescripcionPeriodo+Formateo(DameAño(
                    PresupuestoMes.FechaFinal)),Verdadero,Verdadero))
      Excell_Pinta(2,i,'EMITIDO',Verdadero,Verdadero)
      Excell_Pinta(2,i+1,'PEDIDO',Verdadero,Verdadero)
      Excell_Pinta(2,i+2,'CARTERA',Verdadero,Verdadero)
      Excell_Pinta(2,i+3,'TOTAL',Verdadero,Verdadero)
      NumColumna=i
       Excell_Columnas
       ColDesde=xlColID
       NumColumna=i+3
       colHasta=xlColID
       HojaExcel.Rango(colDesde+'1:'+colHasta+'1').Combinar
       HojaExcel.Rango(colDesde+'1:'+colHasta+'1').Alinear
       i=i+4
    Hasta registro PresupuestoMes sea vacío
Excell_Pinta(1,i,'TOTALES',Verdadero,Verdadero)
Excell_Pinta(2,i,'EMITIDO',Verdadero,Verdadero)
Excell_Pinta(2,i+1,'PEDIDO',Verdadero,Verdadero)
Excell_Pinta(2,i+2,'CARTERA',Verdadero,Verdadero)
Excell_Pinta(2,i+3,'TOTAL',Verdadero,Verdadero)
NumColumna=i
Excell_Columnas
ColDesde=xlColID
NumColumna=i+3
colHasta=xlColID
HojaExcel.Rango(colDesde+'1:'+colHasta+'1').Combinar
HojaExcel.Rango(colDesde+'1:'+colHasta+'1').Alinear
fila=2
```

```
DetallePresup se inicializa
DetallePresup se filtra CodProveedor sea Proveedor.No)
DetallePresup se filtra Tipo sea <>Pagado
Si no encuentra registros DetallePresupuesto entonces
     Saltar registro
fila=fila+1
Excell_Pinta(fila,1,Prov.No,Falso,Falso)
Excell_Pinta(fila,2,Prov.Nombre,Falso,Falso)
nvImportePedido=0
DetallePresup se inicializa
DetallePresup se filtra Cod.Proveedor sea Prov.No
DetallePresup se filtra Tipo sea Pedido O Reservado
DetallePresup se filtra FechaVcto sea <=optFechaAgrup
Si encuentra registros DetallePresup entonces
    Repetir
       nvImportePedido=nvImportePedido+DetallePresup.Importe
    Hasta registro DetallePresup sea vacío
Excell_Pinta(fila,4,Formateo(nvImportePedido),Falso,Falso)
DetallePresup se inicializa
DetallePresup se filtra Cod.Proveedor sea Prov.No
DetallePresup se filtra Tipo sea Cartera
DetallePresup se filtra FechaVcto sea <=optFechaAgrup
Si encuentra registros DetallePresup entonces
   Repetir
      Si SituacionRemOrdPagRegis entonces
         nvImporteEmitido=nvImporteEmitido+DetallePresup.Importe
      Sino Si PagareEmitido entonces
         nvImporteEmitido=nvImporteEmitido+DetallePresup.Importe
     Sino Si FPagoCTODOC entonces
        nvImporteEmitido=nvImporteEmitido+DetallePresup.Importe
     Sino
        nvImporteCartera=nvImporteCartera+DetallePresup.Importe
   Hasta registro DetallePresup sea vacío
Excell_Pinta(fila,3,Formateo(nvImporteEmitido),Falso,Falso)
Excell_Pinta(fila,5,Formateo(nvImporteCartera),Falso,Falso)
Formula='='+'SUMA('+'C'+Formateo(fila)+':'+'E'+Formateo(Fila)+')'
Excell_Pinta(fila,6,Formula,Falso,Falso)
i=7
PresupMes se inicializa
PresupMes se filtra TipoPresupuesto sea Proveedores
PresupMes se filtra FechaInicial sea mayor optFechaAgrup
Si encuentra registros PresupMes entonces
    Repetir
       nvImportePedido=0
       nvImporteEmitido=0
       nvImporteCartera=0
       DetallePresup se inicializa
       DetallePresup se filtra CodProveedor sea Prov.No
       DetallePresup se filtra Tipo sea Pedido O Reservado
       DetallePresup se filtra FechaVencimiento sea  >=PresupMes.FechaInicial y sea <=
       PresupMes.FechaFinal
       Si encuentra registros DetallePresup entonces
           Repetir
              nvImportePedido=nvImportePedido+DetallePresup.Importe
```

```
        Hasta registro DetallePresup sea vacío
    Excell_Pinta(fila,i+1,Formateo(nvImportePedido),Falso,Falso)
    DetallePresup se inicializa
    DetallePresup se filtra CodProveedor sea Prov.No
    DetallePresup se filtra Tipo sea Cartera
    DetallePresup se filtra FechaVencimiento sea  >=PresupMes.FechaInicial y sea <=
    PresupMes.FechaFinal
    Si encuentra registros DetallePresup entonces
       Repetir
          Si SituacionRemOrdPagRegis entonces
             nvImporteEmitido=nvImporteEmitido+DetallePresup.Importe
          Sino Si PagareEmitido entonces
            nvImporteEmitido=nvImporteEmitido+DetallePresup.Importe
          Sino Si FPagoCTODOC entonces
            nvImporteEmitido=nvImporteEmitido+DetallePresup.Importe
          Sino
            nvImporteCartera=nvImporteCartera+DetallePresup.Importe
       Hasta registro DetallePresup sea vacío
       Excell_Pinta(fila,i,Formateo(nvImporteEmitido),Falso,Falso)
       colDesde=xlColdID
       Excell_Pinta(fila,i+2,Formateo(nvImporteCartera),Falso,Falso)
       colHasta=xlColID
       Formula='='+'SUMA('+coldesde+Formateo(fila)+':'+colHasta+
                  Formateo(fila)+')'
       Excell_Pinta(fila,i+3,Formula,Falso,Falso)
       i=i+4
Hasta registro PresupMes sea vacío
Formula='='
For aux=3 hasta i-1 hacer
    NumColumna=aux
    Excell_Columnas()
    Formula=Formula+'+'+xlColID+Formateo(Fila)
    aux=aux+3
Excell_Pinta(fila,aux+1,Formula,Verdadero,Falso)
Formula='='
For aux=4 hasta i-1 hacer
    NumColumna=aux
    Excell_Columnas()
    Formula=Formula+'+'+xlColID+Formateo(Fila)
    aux=aux+3
Excell_Pinta(fila,aux+1,Formula,Verdadero,Falso)
Formula='='
For aux=5 hasta i-1 hacer
  NumColumna=aux
  Excell_Columnas()
  Formula=Formula+'+'+xlColID+Formateo(Fila)
  aux=aux+3
Excell_Pinta(fila,aux+1,Formula,Verdadero,Falso)
Formula='='
For aux=6 hasta i-1 hacer
  NumColumna=aux
  Excell_Columnas()
  Formula=Formula+'+'+xlColID+Formateo(Fila)
  aux=aux+3
```

```
    Excell_Pinta(fila,aux+1,Formula,Verdadero,Falso)
    fila=fila+2
    i=i+3
   For aux=3 to i hacer
     Excell_Pinta(fila,aux,’’,Falso,Falso)
     Formula=’=’+’SUMA(‘+xColID+Formateo(3)+’:’+xColID+Formateo(fila-2)+’)’
     Excell_Pinta(fila,aux,Formula,Falso,Falso)
  HojaExcel.Rango(B:B).AutoAjustar
  HojaExcel.Rango(C3).Seleccionar
  AplicacionExcel.Bloquear=Verdadero
```

Código Original

```
IF optFechaAgrup = 0D THEN
  ERROR('Debes seleccionar la fecha de agrupación del primer vencimiento');

IF NOT CREATE(ApllExcel,TRUE) THEN
   ERROR('ERROR AL ACCEDER A EXCELL.\'+
       'Se ha producido un error al instanciar Excell');

ApllExcel.Visible(TRUE);
LibroExcel := ApllExcel.Workbooks.Add(-4167);
HojaExcel := LibroExcel.Worksheets.Item('Hoja1');
HojaExcel.Activate;
HojaExcel.Name:='Pres.Compra';

Excell_Pinta(2,1,'Cod.',TRUE,TRUE);
Excell_Pinta(2,2,'Proveedor',TRUE,TRUE);

RangoFechas.RESET;
RangoFechas.SETRANGE("Tipo                                    Presupuesto",RangoFechas."Tipo
Presupuesto"::Proveedores);
RangoFechas.SETFILTER(RangoFechas."Fecha Inicial",'> %1',optFechaAgrup);

i:=3;
Excell_Pinta(1,i,'Vencido a 31/12/2009',TRUE,TRUE);
HojaExcel.Range('C1:F1').Merge;
HojaExcel.Range('C1:F1').HorizontalAlignment:= -4108;

Excell_Pinta(2,i,'EMITIDO',TRUE,TRUE);
Excell_Pinta(2,i+1,'PEDIDO',TRUE,TRUE);
Excell_Pinta(2,i+2,'CARTERA',TRUE,TRUE);
Excell_Pinta(2,i+3,'TOTAL',TRUE,TRUE);
i+=4;

IF RangoFechas.FINDSET THEN
  REPEAT
    IF i<> 2 THEN
       Excell_Pinta(1,i,RangoFechas."Descripción
Periodo"+FORMAT(DATE2DMY(RangoFechas."fecha final",3)),TRUE,TRUE);

    Excell_Pinta(2,i,'EMITIDO',TRUE,TRUE);
    Excell_Pinta(2,i+1,'PEDIDO',TRUE,TRUE);
    Excell_Pinta(2,i+2,'CARTERA',TRUE,TRUE);
    Excell_Pinta(2,i+3,'TOTAL',TRUE,TRUE);
```

```
    NumColumna:=i;
    Excell_Columnas();
    colDesde:=xlColID;

    NumColumna:=i+3;
    Excell_Columnas();
    colHasta:=xlColID;

    HojaExcel.Range(colDesde + '1:' + colHasta + '1').Merge;
    HojaExcel.Range(colDesde + '1:' + colHasta + '1').HorizontalAlignment:= -4108;
    i+=4;
  UNTIL RangoFechas.NEXT=0;

// TOTALES DE TOTALES
Excell_Pinta(1,i,'TOTALES',TRUE,TRUE);
Excell_Pinta(2,i,'EMITIDO',TRUE,TRUE);
Excell_Pinta(2,i+1,'PEDIDO',TRUE,TRUE);
Excell_Pinta(2,i+2,'CARTERA',TRUE,TRUE);
Excell_Pinta(2,i+3,'TOTAL',TRUE,TRUE);

NumColumna:=i;
Excell_Columnas();
colDesde:=xlColID;

NumColumna:=i+3;
Excell_Columnas();
colHasta:=xlColID;

HojaExcel.Range(colDesde + '1:' + colHasta + '1').Merge;
HojaExcel.Range(colDesde + '1:' + colHasta + '1').HorizontalAlignment:= -4108;

fila:=2;

rcdDetallePresup.RESET;
rcdDetallePresup.SETRANGE("Cód. Proveedor",Vendor."No.");
rcdDetallePresup.SETFILTER(Tipo,'<>%1',rcdDetallePresup.Tipo::Pagado);

IF NOT rcdDetallePresup.FINDFIRST THEN
  CurrReport.SKIP;

fila+=1;
Excell_Pinta(fila,1,Vendor."No.",FALSE,FALSE);
Excell_Pinta(fila,2,Vendor.Name,FALSE,FALSE);

nvImportePedido:=0;
rcdDetallePresup.RESET;
rcdDetallePresup.SETRANGE("Cód. Proveedor",Vendor."No.");
rcdDetallePresup.SETFILTER(Tipo,'%1|%2',rcdDetallePresup.Tipo::Pedido,rcdDetallePresup.T
ipo::Reservado);
rcdDetallePresup.SETFILTER("Fecha Vencimiento",'<= %1',optFechaAgrup);
IF rcdDetallePresup.FINDSET THEN
REPEAT
  nvImportePedido+=rcdDetallePresup.Importe;
```

```
UNTIL rcdDetallePresup.NEXT=0;

Excell_Pinta(fila,4,FORMAT(nvImportePedido),FALSE,FALSE);

rcdDetallePresup.RESET;
rcdDetallePresup.SETRANGE("Cód. Proveedor",Vendor."No.");
rcdDetallePresup.SETRANGE(Tipo,rcdDetallePresup.Tipo::Cartera);
rcdDetallePresup.SETFILTER("Fecha Vencimiento",'<= %1',optFechaAgrup);
IF rcdDetallePresup.FINDSET THEN
REPEAT
  IF SituacionRemOrdPagRegis() THEN
    nvImporteEmitido+=rcdDetallePresup.Importe
  ELSE IF PagareEmitido() THEN
    nvImporteEmitido+=rcdDetallePresup.Importe
  ELSE IF FPagoCTODOC() THEN
    nvImporteEmitido+=rcdDetallePresup.Importe
  ELSE
    nvImporteCartera+=rcdDetallePresup.Importe;
UNTIL rcdDetallePresup.NEXT=0;

Excell_Pinta(fila,3,FORMAT(nvImporteEmitido),FALSE,FALSE);
Excell_Pinta(fila,5,FORMAT(nvImporteCartera),FALSE,FALSE);

// Total
Formula:='='+'SUMA('+'C'+FORMAT(fila) + ':' + 'E'+FORMAT(fila)+')';
Excell_Pinta_Formula(fila,6,Formula,FALSE,FALSE,6);


i:=7;

RangoFechas.RESET;
RangoFechas.SETRANGE("Tipo                                Presupuesto",RangoFechas."Tipo
Presupuesto"::Proveedores);
RangoFechas.SETFILTER(RangoFechas."Fecha Inicial",'> %1',optFechaAgrup);
IF RangoFechas.FINDSET THEN
REPEAT

  nvImportePedido:=0;
  nvImporteEmitido:=0;
  nvImporteCartera:=0;

  rcdDetallePresup.RESET;
  rcdDetallePresup.SETRANGE("Cód.                                Proveedor",Vendor."No.");
rcdDetallePresup.SETFILTER(Tipo,'%1|%2',rcdDetallePresup.Tipo::Pedido,rcdDetallePresup.T
ipo::Reservado);
  rcdDetallePresup.SETRANGE("Fecha                          Vencimiento",RangoFechas."Fecha
Inicial",NORMALDATE(RangoFechas."fecha final"));
  IF rcdDetallePresup.FINDSET THEN
  REPEAT
    nvImportePedido+=rcdDetallePresup.Importe;
  UNTIL rcdDetallePresup.NEXT=0;

  Excell_Pinta(fila,i+1,FORMAT(nvImportePedido),FALSE,FALSE);
```

```
  rcdDetallePresup.RESET;
  rcdDetallePresup.SETRANGE("Cód. Proveedor",Vendor."No.");
  rcdDetallePresup.SETRANGE(Tipo,rcdDetallePresup.Tipo::Cartera);
  rcdDetallePresup.SETRANGE("Fecha                    Vencimiento",RangoFechas."Fecha
Inicial",NORMALDATE(RangoFechas."fecha final"));
  IF rcdDetallePresup.FINDSET THEN
  REPEAT
    IF SituacionRemOrdPagRegis() THEN
      nvImporteEmitido+=rcdDetallePresup.Importe
    ELSE IF PagareEmitido() THEN
      nvImporteEmitido+=rcdDetallePresup.Importe
    ELSE IF FPagoCTODOC() THEN
      nvImporteEmitido+=rcdDetallePresup.Importe
    ELSE
      nvImporteCartera+=rcdDetallePresup.Importe;
  UNTIL rcdDetallePresup.NEXT=0;

  Excell_Pinta(fila,i,FORMAT(nvImporteEmitido),FALSE,FALSE);
  colDesde:=xlColID;
  Excell_Pinta(fila,i+2,FORMAT(nvImporteCartera),FALSE,FALSE);
  colHasta:=xlColID;

  // Total
  Formula:='='+'SUMA(' + colDesde + FORMAT(fila) + ':' + colHasta + FORMAT(fila)+')';
  Excell_Pinta_Formula(fila,i+3,Formula,FALSE,FALSE,6);
  i+=4;
UNTIL RangoFechas.NEXT=0;

// Pintamos los totales por proveedor del EMITIDO
Formula:='=';
FOR aux:=3 TO i-1 DO
BEGIN
  // Me situo en la celda en cuestión para inicializar xlColID
  NumColumna:=aux;
  Excell_Columnas();
  Formula+='+' + xlColID + FORMAT(fila);
  aux+=3;
END;
Excell_Pinta_Formula(fila,aux+1,Formula,TRUE,FALSE,15);

// Pintamos los totales por proveedor del PEDIDO
Formula:='=';
FOR aux:=4 TO i-1 DO
BEGIN
  // Me situo en la celda en cuestión para inicializar xlColID
  NumColumna:=aux;
  Excell_Columnas();
  Formula+='+' + xlColID + FORMAT(fila);
  aux+=3;
END;
Excell_Pinta_Formula(fila,aux+1,Formula,TRUE,FALSE,15);

// Pintamos los totales por proveedor del CARTERA
Formula:='=';
```

```
FOR aux:=5 TO i-1 DO
BEGIN
  // Me situo en la celda en cuestión para inicializar xlColID
  NumColumna:=aux;
  Excell_Columnas();
  Formula+='+' + xlColID + FORMAT(fila);
  aux+=3;
END;
Excell_Pinta_Formula(fila,aux+1,Formula,TRUE,FALSE,15);

// Pintamos los totales por proveedor del TOTAL
Formula:='=';
FOR aux:=6 TO i-1 DO
BEGIN
  // Me situo en la celda en cuestión para inicializar xlColID
  NumColumna:=aux;
  Excell_Columnas();
  Formula+='+' + xlColID + FORMAT(fila);
  aux+=3;
END;
Excell_Pinta_Formula(fila,aux+1,Formula,TRUE,FALSE,15);
fila+=2;
i+=3;
FOR aux:=3 TO i DO
BEGIN
  // Me situo en la celda en cuestión para inicializar xlColID y xlRowID
  Excell_Pinta(fila,aux,'',FALSE,FALSE);
  // Genero el sumatorio de toda la columna

  Formula:='='+'SUMA(' + xlColID + FORMAT(3) + ':' + xlColID + FORMAT(fila-2)+')';
  Excell_Pinta_Formula(fila,aux,Formula,TRUE,FALSE,6);
END;
HojaExcel.Range('B:B').Columns.AutoFit;
HojaExcel.Range('C3').Select;
AplExcel.ActiveWindow.FreezePanes := TRUE;
```

SituacionRemOrdPagRegis

PseudoCódigo

```
Si DetallePresup.NoDocumento=''
    Salir(Falso)
TablaEmpresa se inicializa
Si encuentra registros TablaEmpresa entonces
  Repetir
     MovProv se inicializa
     MovProv.CambioEmpresa(TablaEmpresa.Nombre)
     MovProv se filtra NoDocumento sea DetallePresup.NoDocumento
     MovProv se filtra NoFactura sea DetallePresup.NoEfecto
     Si encuentra registros MovProv entonces
          Si MovProv.SituacionDocumento=Registrado
              Salir(Verdadero)
  Hasta registro TablaEmpresa sea vacío
Salir(Falso)
```

Código Original

```
IF rcdDetallePresup."Nº Documento" =" THEN
  EXIT(FALSE);
rcdEmpresa.RESET;
IF rcdEmpresa.FINDSET THEN
REPEAT
 rcdMovProv.RESET;
 rcdMovProv.CHANGECOMPANY(rcdEmpresa.Name);
 rcdMovProv.SETCURRENTKEY("Document Type","Document No.","Vendor No.");
 rcdMovProv.SETRANGE("Document No.",rcdDetallePresup."Nº Documento");
 rcdMovProv.SETRANGE("Bill No.",rcdDetallePresup."Nº Efecto");
 IF rcdMovProv.FINDFIRST THEN
 BEGIN
   IF rcdMovProv."Document Situation"=rcdMovProv."Document Situation"::"Posted BG/PO"
THEN
     EXIT(TRUE);
 END;
UNTIL (rcdEmpresa.NEXT=0);
EXIT(FALSE);
```

PagareEmitido

PseudoCódigo

```
Si DetallePresup.NoDocumento=''
    Salir(Falso)
TablaEmpresa se inicializa
Si encuentra registros TablaEmpresa entonces
 Repetir
    MovProv se inicializa
    MovProv.CambioEmpresa(TablaEmpresa.Nombre)
    MovProv se filtra NoDocumento sea DetallePresup.NoDocumento
    MovProv se filtra NoFactura sea DetallePresup.NoEfecto
    MovProv se filtra SituacionDocumento sea Cartera
    Si encuentra registros MovProv entonces
        MovProv calcula PagareEmitido
        Si MovProv.PagareEmitido
            Salir(Verdadero)
 Hasta registro TablaEmpresa sea vacío
Salir(Falso)
```

Código Original

```
IF rcdDetallePresup."Nº Documento" =" THEN
  EXIT(FALSE);
rcdEmpresa.RESET;
IF rcdEmpresa.FINDSET THEN
REPEAT
 rcdMovProv.RESET;
 rcdMovProv.CHANGECOMPANY(rcdEmpresa.Name);
 rcdMovProv.SETCURRENTKEY("Document Type","Document No.","Vendor No.");
 rcdMovProv.SETRANGE("Document No.",rcdDetallePresup."Nº Documento");
 rcdMovProv.SETRANGE("Bill No.",rcdDetallePresup."Nº Efecto");
```

```
  rcdMovProv.SETRANGE("Document                    Situation",rcdMovProv."Document
Situation"::Cartera);
  IF rcdMovProv.FINDFIRST THEN
  BEGIN
    rcdMovProv.CALCFIELDS("Pagare emitido");
    IF rcdMovProv."Pagare emitido" THEN
      EXIT(TRUE);
  END;
UNTIL (rcdEmpresa.NEXT=0);
EXIT(FALSE);
```

FPagoCtoDoc

PseudoCódigo

```
Si DetallePresup.NoDocumento=''
    Salir(Falso)
TablaEmpresa se inicializa
Si encuentra registros TablaEmpresa entonces
  Repetir
     MovProv se inicializa
     MovProv.CambioEmpresa(TablaEmpresa.Nombre)
     MovProv se filtra NoDocumento sea DetallePresup.NoDocumento
     MovProv se filtra NoFactura sea DetallePresup.NoEfecto
     MovProv se filtra SituacionDocumento sea Cartera
     Si encuentra registros MovProv entonces
        MovProv calcula FormaPagoDocCartera
        Si MovProv.FormaPagoDocCartera='PAGOFINANC'
           Salir(Verdadero)
  Hasta registro TablaEmpresa sea vacío
Salir(Falso)
```

Código Original

```
IF rcdDetallePresup."Nº Documento" =" THEN
   EXIT(FALSE);
rcdEmpresa.RESET;
IF rcdEmpresa.FINDSET THEN
REPEAT
  rcdMovProv.RESET;
  rcdMovProv.CHANGECOMPANY(rcdEmpresa.Name);
  rcdMovProv.SETCURRENTKEY("Document Type","Document No.","Vendor No.");
  rcdMovProv.SETRANGE("Document No.",rcdDetallePresup."Nº Documento");
  rcdMovProv.SETRANGE("Bill No.",rcdDetallePresup."Nº Efecto");
  rcdMovProv.SETRANGE("Document                    Situation",rcdMovProv."Document
Situation"::Cartera);
  IF rcdMovProv.FINDFIRST THEN
  BEGIN
    rcdMovProv.CALCFIELDS(rcdMovProv."Forma pago en Doc. cartera");
      IF rcdMovProv."Forma pago en Doc. cartera" = 'PAGOFINANC' THEN
       EXIT(TRUE);
  END;
UNTIL (rcdEmpresa.NEXT=0);
EXIT(FALSE);
```

Reasignar

PseudoCódigo

```
PresupMes.Coge(TablaMes.NoPptoAño,TablaMes.NoPptoMes)
Si Confirma(‘Estas seguro que deseas reasignar los % de %1’,Falso,
                PresupMes.DescripcionPeriodo) entonces
   DetallePresup se inicializa
   DetallePresup se filtra NoPptoAño sea TablaMes.NoPptoAño
   DetallePresup se filtra NoPptoMes sea TablaMes.NoPptoMes
   DetallePresup.ModificarTodos(% s/Presupuesto Mes,0,Verdadero)
   DetallePresup.ModificarTodos(ImportePresupuestado,0)
```

Código Original

```
IF rcdMes.GET("Nº Ppto Año","Nº Ppto Mes") THEN;
IF CONFIRM('Estas seguro que deseas reasignar los % de %1',FALSE,rcdMes."Descripción
Periodo") THEN
BEGIN
  rcdDetalle.RESET;
  rcdDetalle.SETRANGE("Nº Ppto Año","Nº Ppto Año");
  rcdDetalle.SETRANGE("Nº Ppto Mes","Nº Ppto Mes");
  rcdDetalle.MODIFYALL("% s/Presupuesto Mes",0,TRUE);
  rcdDetalle.MODIFYALL("Importe Presupuestado",0);
END;
```

4.2.5 Tienda

En este apartado, se explica toda la funcionalidad referente a la parte que utilizan las tiendas para poder vender, abonar y reservar productos. Además, se comenta cómo realizan pedidos a central, cómo realizan la apertura y el cierre del día y cómo se realiza la devolución de productos a Central ya sea por exceso, rotura, etc.

Apertura del día (TablaCaja)

En este caso, la apertura del día es lo primero que se hace nada más abrir la tienda. En caso de no realizar la apertura, el sistema no deja hacer ninguna operación, ya que no sabe a que día debe imputarla.

Al abrir la pantalla de apertura de día, el sistema comprueba si se ha hecho alguna apertura en ese día. Si se cumple dicho requisito, el sistema emite un mensaje de error advirtiendo de ese hecho. Una vez hechas las comprobaciones, el personal de tienda debe registrar el inicio del día.

El sistema crea un nuevo registro de caja, para que el personal de tienda pueda realizar cualquier operación.

Ilustración 57 Pantalla donde se realiza la apertura de la caja

Abrir Pantalla

PseudoCódigo

```
ConfigTPV se posiciona en el primer registro
Numero=ConfigTPV.NumCaja
TablaCaja se inicializa
TablaCaja se filtra CodTienda sea ConfigTPV.CodTienda
TablaCaja se filtra CodTpv sea Numero
TablaCaja se filtra Apertura sea Verdadero
Si encuentra registro TablaCaja entonces
   Si TablaCaja.Activa entonces
      ERROR('Para poder abrir la caja de hoy primero ha de cerrar la caja del día %1 y
            turno %2.',TablaCaja.FechaInicio,TablaCaja.NoTurno)
```

Código Original

```
ConfigTPV.FIND('-');
NTpv := cduILR.DameTPV();
//COMPROBAMOS QUE LA ULTIMA CAJA ESTA CERRADA
CajaActiva.RESET;
CajaActiva.SETRANGE("Cód. tienda",ConfigTPV."Cód. tienda");
CajaActiva.SETRANGE("Cód. TPV",NTpv);
CajaActiva.SETRANGE(Apertura,TRUE);
IF CajaActiva.FIND('+') THEN
BEGIN
  IF CajaActiva.Activa THEN
    ERROR(Text00001,CajaActiva."Fecha inicio día",CajaActiva."Nº turno");
END;
```

Registrar Apertura(TablaCaja)

PseudoCódigo

```
TablaCaja se filtra CodTpv sea Numero
TablaCaja se filtra Activa sea Verdadero
Si no encuentra registros TablaCaja entonces
     Insertar Registro TablaCaja
Si No Confirmas('Está seguro de realizar la apertura del tpv?',Falso)
     Salir
Si No Apertura entonces
   Si SaldoInicial=0 entonces
      ERROR('Ha de introducir el saldo inicial del tpv.')
  Si FechaInicio<>HOY entonces
      ERROR('La fecha del inicio de día no es correcta')
  TablaCaja.FechaUltimaMod=HOY
  TablaCaja.SaldoInicial=SaldoInicial
  TablaCaja.Apertura=Verdadero
  Modificar registro TablaCaja
Sino
   ERROR(''La apertura de la tienda %1, tpv %2, día %3 y turno %4 ya ha sido
            realizada.', TablaCaja.CodTienda,TablaCaja.CodTpv,
           TablaCaja.FechaInicio, TablaCaja.NoTurno)
Si TablaCaja.NoTurno=1 entonces
    TablaCaja.ErrorEnvio=Verdadero
    Modificar registro TablaCaja
    Si ServicioWeb.TPV_Cierre_Caja(ConfigTPV.CodTienda,ConfigTPV.Password,
                                   TablaCaja) entonces
       TablaCaja.ErrorEnvio=Falso
       Modificar registro TablaCaja
       ConfigTPV.NoFallos=0
       ConfigTPV.ModoTrabajo=Online
       Modificar registro ConfigTPV
Si TablaCaja.ErrorEnvio entonces
    Message('Apertura registrada')
Sino
    Si Confirmar('¿Desea sincronizar con la CENTRAL ahora?')
       ConfigTpv se posiciona primer registro
       ServicioWeb.ST_Abrir_Login(ConfigTpv.CodTienda,ConfigTpv,Password,
                                  Falso)
       Contador=ServicioWeb.TPV_Sincronizar_Vendedores(ConfigTpv.CodTienda,
                ConfigTpv.Password, Formateo(ConfigTpv.FechaUltimaMod),Hoy)
       Si Contador>0 entonces
          Contador=0
          ConfigTpv.UltimaFechaMod=Hoy
          Modificar registro ConfigTpv
        Contador=ServicioWeb.TPV_Sincronizar_Tiendas(ConfigTpv.CodTienda,
                ConfigTpv.Password)
       Si Contador>0 entonces
          Contador=0
          ConfigTpv.UltimaFechaMod=Hoy
          Modificar registro ConfigTpv
       Contador=ServicioWeb.TPV_Sincronizar_Transportistas(ConfigTpv.CodTienda,
                ConfigTpv.Password)
```

```
Si Contador>0 entonces
   Contador=0
   ConfigTpv.UltimaFechaMod=Hoy
   Modificar registro ConfigTpv
Contador=0
dvFecha=ConfigTpv.FechaUltimaMod
numDias=Hoy-dvFecha
Si numDias=0 entonces
   numDias=1
Mientras dvFecha<=Hoy hacer
    Contador=ServicioWeb.TPV_Sincronizar_Productos(ConfigTpv.Codtienda,
                ConfigTpv.Password,dvFecha,dvfecha)
    ConfigTpv.UltimaModFechaProd=dvfecha
    Modificar registro ConfigTpv
    dvFecha=Calcular('1D',dvFecha)
TablaCaja.Sincronizado=Verdadero
TablaCaja.FechaSincro=Hoy
TablaCaja.HoraSincro=Tiempo
```

Código Original

```
Rec.SETFILTER("Cód. TPV",NTpv);
Rec.SETRANGE(Activa,TRUE);
IF NOT Rec.FIND('+') THEN
BEGIN
   INSERT(TRUE);
END;
FILTERGROUP(2);
CurrForm."Nº turno".ACTIVATE;
CurrForm.LANGUAGE(cduILR.ActivarIdioma());

IF NOT CONFIRM(Text00001,FALSE) THEN
  EXIT;
IF NOT Apertura THEN BEGIN
  IF SaldoIni = 0 THEN
  BEGIN
    ERROR(Text00002);
  END;
  IF "Fecha inicio día" <> WORKDATE THEN
     ERROR(Text00003);
  "Fecha últ. modificación":= WORKDATE;
  "Saldo inicial" := SaldoIni;
  Apertura := TRUE;
  MODIFY;
  COMMIT;
END ELSE BEGIN
  ERROR(Text00004,"Cód. tienda","Cód. TPV","Fecha inicio día","Nº turno")
END;
IF "Nº turno" = 1 THEN
BEGIN
   "Error Envío":=TRUE;
   MODIFY;
   IF ws.TPV_Cierre_Caja(ConfigTPV."Cód. tienda",ConfigTPV.Password,Rec) THEN
   BEGIN
     "Error Envío":=FALSE;
```

```
    MODIFY;
    ConfigTPV."Nº de fallos":=0;
    ConfigTPV."Modo de Trabajo":=ConfigTPV."Modo de Trabajo"::"On-Line";
    ConfigTPV.MODIFY;
  END;
END;
IF "Error Envío" THEN
   MESSAGE(Text00005)
ELSE BEGIN
   IF CONFIRM(Text00006) THEN BEGIN
      Conf.FINDFIRST;
      ws.ST_Abrir_Login(Conf."Cód. tienda",Conf.Password,FALSE);
      Contador:=ws.TPV_Sincronizar_Vendedores(Conf."Cód.
tienda",Conf.Password,FORMAT(Conf."Ult. Fecha Actualiz. Vend."),
                          FORMAT(WORKDATE));
      IF Contador > 0  THEN BEGIN
        Contador:=0;
        Conf."Ult. Fecha Actualiz. Vend.":=WORKDATE;
        Conf.MODIFY;
      END;
     Contador:= ws.TPV_Sincronizar_Tiendas(Conf."Cód. tienda",Conf.Password);
      IF Contador > 0  THEN BEGIN
        Contador:=0;
        Conf."Ult. Fecha Actualiz. Tiendas":=WORKDATE;
      END;
     Contador:=ws.TPV_Sincronizar_Transportistas(Conf."Cód. tienda",Conf.Password);
      IF Contador > 0  THEN BEGIN
        Contador:=0;
      END;
     Contador:=0;
     IndProductos:=0;
     dvFecha:= Conf."Ult. Fecha Actualiz. Prod.";
     numDias:= WORKDATE - dvFecha;
     IF numDias = 0 THEN
       numDias:=1;
     WHILE dvFecha <= WORKDATE DO
     BEGIN
      Contador+=ws.TPV_Sincronizar_Productos(Conf."Cód.
tienda",Conf.Password,FORMAT(dvFecha),FORMAT(dvFecha),Ven);
      Conf."Ult. Fecha Actualiz. Prod.":=dvFecha;
      Conf.MODIFY;
      COMMIT;
      dvFecha := CALCDATE('1D',dvFecha);
     END;
      "Sincronizado CENTRAL":=TRUE;
      "Fecha Sincronización":=WORKDATE;
      "Hora  Sincronización":= TIME;
     CurrForm.UPDATE(TRUE);
   END;
END;
cduTcikets.ProcesarFichero();
CurrForm.CLOSE;
```

Fin Día

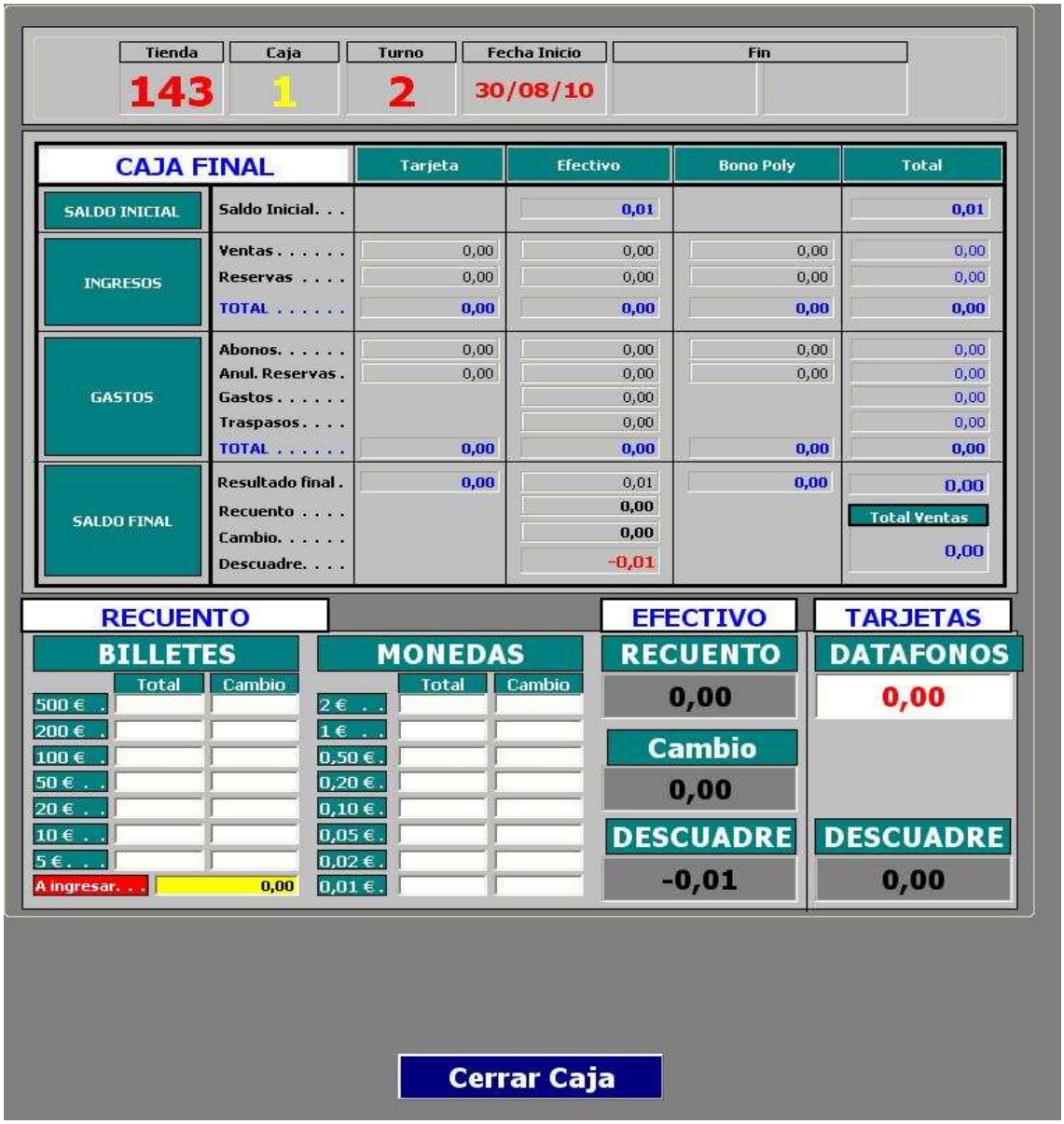


Ilustración 58 Pantalla de fin de día, donde deben rellenar el recuento de billetes y monedas

El responsable de la tienda debe entrar en esta pantalla(ilustración 58) para realizar el cierre del día. Como se observa en la pantalla, se indica el valor que se ha generado para cada una de las formas de pago, tanto si es venta como si es abono. A partir de esos datos, el sistema calcula el Total de Ventas que ha realizado la tienda en el día en que corresponde.

Entonces, el responsable debe rellenar la parte de recuento, donde indica la cantidad de billetes o monedas que tiene en ese momento en el cajon. Por defecto, cada vez que

inserta una cantidad, el sistema lo pone todo para cambio y el responsable de tienda debe decidir qué dejar para cambio y qué ingresar en el banco.

Una vez rellenados todos esos campos, el sistema calcula el descuadre entre lo que ha recontado y lo que ha hecho la tienda en el de día que corresponde y lo muestra para que el responsable de tienda, en caso de haber descuadre, revise si existe algún error. Por otro lado, debe hacer el cierre de los datafonos para que pueda indicar el recuento de tarjeta. Una vez indicados los campos necesarios y haber hecho las pertinentes revisiones, realiza el cierre. Entonces, el sistema hace un movimiento de traspaso para indicar que se ha realizado el cierre. Además, imprime una tira de caja con todos los datos que se mostraban en pantalla para tener un historial de cierres de caja

Abrir Pantalla(TablaCaja)

PseudoCódigo

```
TablaCaja se filtra CodTpv sea Numero
TablaCaja se filtra Activa sea Verdadero
Si encuentra registros TablaCaja entonces
   CabVenta se inicializa
   CabVenta se filtra TipoDocumento sea Oferta
   Si encuentra registro CabVenta entonces
       Borrar todos los registros CabVenta
   CuantoDoc=0
   CabVenta se inicializa
   CabVenta se filtra TPV sea Verdadero
   CabVenta se filtra FechRegistro sea <>Hoy
   Si encuentra registros CabVenta entonces
      CuantosDoc=CabVenta.Contar
      Si  No Confirma('¡¡¡ ATENCION !!!\, Tienes %1 tickets sin facturar.\,
                          ¿Deseas continuar?',Falso,CuantosDoc) entonces
         ERROR('Proceso cancelado')
   TablaBonos se inicializa
   TablaBonos se filtra ErrorEnvio sea Falso
   Borrar todos los registros TablaBonos
Sino
   TablaCaja se filtra Activa
   Si no encuentra ultimo registro TablaCaja entonces
      TablaCaja se filtra CodTpv
      Si no encuentra ultimo registro TablaCaja entonces
          ERROR('No se ha encontrado ningun arqueo de caja. Debes primero haber
                  realizado el inicio de día ')
```

Código Original

```
CurrForm.LANGUAGE(cduILR.ActivarIdioma());
// Por defecto nos posicionamos en la caja activa para este tpv
NTpv := cduILR.DameTPV();
SETRANGE("Cód. TPV",NTpv);
SETRANGE(Activa,TRUE);
IF FIND('-') THEN
```

```
BEGIN
  CabVenta.SETCURRENTKEY("Document Type",TPV,"Posting Date");
  CabVenta.SETRANGE("Document Type",CabVenta."Document Type"::Quote);
  IF CabVenta.FIND('-') THEN
  BEGIN
    CabVenta.DELETEALL();
    COMMIT;
  END;
  CuantosDoc := 0;
  CabVenta.RESET;
  CabVenta.SETCURRENTKEY("Document Type",TPV,"Posting Date");
  CabVenta.SETRANGE(TPV, TRUE);
  CabVenta.SETFILTER("Posting Date", '<>%1',WORKDATE);
  CabVenta.SETRANGE(Totalizado, FALSE);
  IF CabVenta.FIND('-') THEN
  BEGIN
     CuantosDoc:=CabVenta.COUNT;
     IF CuantosDoc > 0 THEN
       IF NOT CONFIRM(Text00001+
                  Text00002+
                  Text00003,FALSE,FORMAT(CuantosDoc)) THEN
         ERROR(Text00004);
  END;
  rcdbonos.RESET;
  rcdbonos.SETRANGE("Error Envio",FALSE);
  rcdbonos.DELETEALL();
END
ELSE
BEGIN
   SETRANGE(Activa);
   IF NOT FIND('+') THEN
   BEGIN
     SETRANGE("Cód. TPV");
     IF NOT FIND('+') THEN
       ERROR(Text00005);
   END;
END;
```

Cerrar Caja(TablaCaja)

PseudoCódigo

```
Si TablaCaja.Registrado entonces
    ERROR('Error, Este arqueo de caja, ya se encuentra registrado')
Si TablaCaja.SaldoFinal=0 entonces
    ERROR('No puedes cerrar caja, sin dejar dinero para cambio')
ImporteTotal=(TablaCaja.SaldoInicial+TablaCaja.VentasEfectivo+
                  TablaCaja.ReservasEfectivo)-(TablaCaja.AbonosEfectivo+
                  TablaCaja.ReservasCanceladasEfectivo+TotalGastos+TotalTraspaso)-
                  TablaCaja.SaldoFinal
TablaCaja.DiferenciaCaja=TablaCaja.TotalRecuento-(TablaCaja.SaldoInicial+
                                TablaCaja.VentasEfectivo+TablaCaja.ReservasEfectivo)-
                                (TablaCaja.AbonosEfectivo+
                                TablaCaja.ReservasCanceladasEfectivo+
                                TablaCaja.TotalGastos+TablaCaja.TotalTraspaso)
```

```
TablaCaja.DiferenciaTarjeta=TablaCaja.RecuentoTarjeta-(TablaCaja.VentasTarjeta+
                                TablaCaja.ReservasTarjeta-TablaCaja.AbonoTarjeta-
                                TablaCaja.ReservasCanceladaTarjeta)
TablaCaja.ImporteIngresar=TablaCaja.TotalRecuento-TablaCaja.SaldoFinal
Si ImporteTotal+TablaCaja.DiferenciaCaja<>0 entonces
    Si No Confirma('¡¡¡¡¡ ATENCION. CIERRE DE CAJA ¡¡¡¡¡\\\ Al registrar el cierre
                    se realizará un traspaso a central por %1?,Falso,ImporteTotal+
                    TablaCaja.DiferenciaCaja) entonces
        Salir
TablaCaja.FechaFin=Today
TablaCaja.HoraFin=Tiempo
Modificar registro TablaCaja
Si ImporteTotal=0 entonces
    EnviarCaja()
Sino
   TablaCaja.DiferenciaCaja=TablaCaja.TotalRecuento-(TablaCaja.SaldoInicial+
                            TablaCaja.VentasEfectivo+TablaCaja.ReservasEfectivo)-
                            (TablaCaja.AbonosEfectivo+
                            TablaCaja.ReservasCanceladasEfectivo+
                            TablaCaja.TotalGastos+TablaCaja.TotalTraspaso)
   TablaCaja.DiferenciaTarjeta=TablaCaja.RecuentoTarjeta-(TablaCaja.VentasTarjeta+
                            TablaCaja.ReservasTarjeta-TablaCaja.AbonoTarjeta-
                            TablaCaja.ReservasCanceladaTarjeta)
   RegTraspaso(ImporteTotal+TablaCaja.DiferenciaCaja,-(TablaCaja.DiferenciaCaja)
   ConfigTpv se inicializa
   ConfigTpv se filtra CodTpv sea Numero
   ConfigTpv se posiciona en el primer registro
   TablaCaja2 se inicializa
   TablaCaja2 se filtra CodTienda sea TablaCaja.CodTienda
   TablaCaja2 se filtra FechaInicio sea TablaCaja.FechaInicio
   TablaCaja2 se filtra HoraInicio sea TablaCaja.HoraInicio
   Caso ConfigTpv.Impresora
        Negra:
          InformeCierreN.UsarOpciones(Falso)
          InformeCierreN.FiltrarTabla(TablaCaja2)
          Ejecutar InformeCierreN
        Blanca:
          InformeCierreB.UsarOpciones(Falso)
          InformeCierreB.FiltrarTabla(TablaCaja2)
          Ejecutar InformeCierreB
TablaCaja.Registrado=Verdadero
TablaCaja.Activa=Falso
Modificar registro TablaCaja
TablaCaja se filtra Activa
Message('Fin de día registrado.')
```

Código Original

```
IF Registrado THEN
  ERROR(Text00001);
IF "Saldo final TPV" = 0 THEN
  ERROR(Text00002);
ImpTotal := ("Saldo inicial"+"Ventas Efectivo"+"Reservas Efectivo")-
        ("Abonos Efectivo"+"Reservas Cancelada Efectivo"+"Total gastos tpv"+"Total
traspasado a central")-
```

```
        ("Saldo final TPV");
 "Diferencia caja":=("Total recuento")-(("Saldo inicial"+"Ventas Efectivo"+"Reservas
Efectivo")-
            ("Abonos Efectivo"+"Reservas Cancelada Efectivo"+"Total gastos tpv"+"Total
traspasado a central"));
"Diferencia tarjeta" := ("Total recuento Tarjeta") -
              ("Ventas Tarjeta"+"Reservas Tarjeta"-"Abonos Tarjeta"-"Reservas Cancelada
Tarjeta");

"Importe a Ingresar" := "Total recuento" - "Saldo final TPV";
IF ImpTotal+"Diferencia caja" <> 0 THEN BEGIN
  IF NOT CONFIRM(Text00003,FALSE, ImpTotal+"Diferencia caja") THEN
    EXIT;
END;
"Fecha fin día" := TODAY;
"Hora fin día" := TIME;
MODIFY;
IF ImpTotal = 0 THEN
BEGIN
  EnviarCaja();
END
ELSE
BEGIN
"Diferencia caja":=("Total recuento")-(("Saldo inicial"+"Ventas Efectivo"+"Reservas
Efectivo")-
            ("Abonos Efectivo"+"Reservas Cancelada Efectivo"+"Total gastos tpv"+"Total
traspasado a central"));

 "Diferencia tarjeta" := ("Total recuento Tarjeta") -
              ("Ventas Tarjeta"+"Reservas Tarjeta"-"Abonos Tarjeta"-"Reservas Cancelada
Tarjeta");

  RegTraspaso(ImpTotal+"Diferencia caja",-"Diferencia caja");
  Config.RESET;
  Config.SETFILTER(Config."Cód. TPV", cduILR.DameTPV());
  Config.FIND('-');
  rcdAuxCaja.SETRANGE("Cód. tienda","Cód. tienda");
  rcdAuxCaja.SETRANGE("Cód. TPV","Cód. TPV");
  rcdAuxCaja.SETRANGE("Fecha inicio día","Fecha inicio día");
  rcdAuxCaja.SETRANGE("Hora inicio día","Hora inicio día");
  CASE Config."Impresora Tickets" OF
      Config."Impresora Tickets"::Negra:
        BEGIN
          // Imprimimos la tira de cierre
          CLEAR(rptCierreNegra);
          rptCierreNegra.USEREQUESTFORM(FALSE);
          rptCierreNegra.SETTABLEVIEW(rcdAuxCaja);
          rptCierreNegra.RUNMODAL;

        END;
      Config."Impresora Tickets"::Blanca:
        BEGIN
          // Imprimimos la tira de cierre
          CLEAR(rptCierreBlanca);
```

```
          rptCierreBlanca.USEREQUESTFORM(FALSE);
          rptCierreBlanca.SETTABLEVIEW(rcdAuxCaja);
          rptCierreBlanca.RUNMODAL;
        END;
  END;
END;
Registrado := TRUE;
Activa := FALSE;
MODIFY;
SETRANGE(Activa);
MESSAGE(Text00004);
COMMIT;
cduILR.EnviarLoPendiente();
cduTcikets.ProcesarFichero();
CurrForm.CLOSE;
```

INGRESOS

. .Tarjeta. .	. Efectivo ..	. Bono . .
Ventas		
0,00	0,00	0,00
Reservas		
0,00	0,00	0,00
Ingresos Totales		
0,00	0,00	0,00

GASTOS

. .Tarjeta. .	. Efectivo ..	. Bono . .
Abonos		
0,00	0,00	0,00
Anulaciones Reservas		
0,00	0,00	0,00
Gastos Tienda		
0,00		
Gastos Totales		
0,00	0,00	0,00

SALDO FINAL

Resultado Tienda . . . . . .0,00€
Total Ventas . . . . . .0,00€

EFECTIVO

Cambio Anterior . . . . . .0,01€
Total Efectivo . . . . . .0,00€

. . . . . .0,01€

Recuento Real . . . . . .0,00€
Descuadre . . . . . .0,00€

A ingresar . . . . . .0,00€
Cambio Final . . . . . .0,00€

TARJETA

Total Ventas Tarj . . . . . .0,00€
Datafonos . . . . . .0,00€
Descuadre . . . . . .0,00€

Firma

Ilustración 59 Ejemplo de impresión Cierre de Caja

RegTraspaso(Importe, Descuadre)

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTienda sea TablaCaja.CodTienda
ConfigTpv se filtra CodTpv sea TablaCaja.CodTpv
ConfigTpv se posiciona en el primer registro
FormaPago.Coge(ConfigTpv.FormaPagoEfectivo)
Si Importe<=0 entonces
    ERROR('Ha de introducir un importe a traspasar.')
MovTraspaso se inicializa
MovTraspaso se filtra TipoMovimiento sea Trasp.Central
MovTraspaso se filtra CodTienda sea TablaCaja.CodTienda
MovTraspaso se filtra CodTpv sea TablaCaja.CodTpv
MovTraspaso se filtra FechaInicio sea TablaCaja.FechaInicio
Si encuentra registros MovTraspaso entonces
    NumDoc=Incrementar(MovTraspaso.NoDocumento)
Sino
    NumDoc=TablaCaja.CodTienda+'/'+TablaCaja.CodTpv+'/'+
                Formateo(TablaCaja.FechaInicio,0,<Year,2><Month,2><Day,2>)+
                '/'+001;
LinDiario se inicializa
LinDiario se filtra NombreDiario sea COBROS
LinDiario se filtra NombreSeccion sea MULTI+TablaCaja.CodTpv
Si encuentra registros LinDiario
    Borrar registros LinDiario
LinDiario se inicializa
LinDiario.NombreDiario='COBROS'
LinDiario.NombreSeccion='MULTI'+TablaCaja.CodTpv
LinDiario.NoLinea=10000
Insertar registro LinDiario
LinDiario.TipoCuenta=Cuenta
LinDiario.NoCuenta=FormaPago.NoCuenta
LinDiario.FechaRegistro=TablaCaja.FechaInicio
LinDiario.TipoDocumento=' '
LinDiario.NoDocumento=NumDoc
LinDiario.Descripcion='Trapaso del Tpv'+TablaCaja.CodTpv+'turno'+
                        TablaCaja.NoTurno+' a la central tienda';
LinDiario.CodigoOrigen='DIACOBROS'
LinDiario.Importe= -(Importe)
LinDiario.CodTienda=TablaCaja.CodTienda
LinDiario.CodTpv=TablaCaja.CodTpv
LinDiario.TipoMovTraspaso=Trasp.Central
LinDiario.NoTurno=TablaCaja.NoTurno
Modificar registro Lin.Diario

Si Descuadre<>0 entonces
   LinDiario se inicializa
   LinDiario.NombreDiario='COBROS'
   LinDiario.NombreSeccion='MULTI'+TablaCaja.CodTpv
   LinDiario.NoLinea=20000
   Insertar registro LinDiario
   LinDiario.TipoCuenta=Cuenta
   LinDiario.NoCuenta=ConfigTpv.CuentaDescuadres
```

```
    LinDiario.FechaRegistro=TablaCaja.FechaInicio
    LinDiario.TipoDocumento=' '
    LinDiario.NoDocumento=NumDoc
    LinDiario.Descripcion='Trapaso del Tpv'+TablaCaja.CodTpv+'turno'+
                          TablaCaja.NoTurno+' a la central tienda';
    LinDiario.CodigoOrigen='DIACOBROS'
    LinDiario.Importe= -(Descuadre)
    LinDiario.CodTienda=TablaCaja.CodTienda
    LinDiario.CodTpv=TablaCaja.CodTpv
    LinDiario.TipoMovTraspaso=Trasp.Central
    LinDiario.NoTurno=TablaCaja.NoTurno
    Modificar registro Lin.Diario

LinDiario se inicializa
LinDiario.NombreDiario='COBROS'
LinDiario.NombreSeccion='MULTI'+TablaCaja.CodTpv
LinDiario.NoLinea=30000
Insertar registro LinDiario
LinDiario.TipoCuenta=Cuenta
LinDiario.NoCuenta=ConfigTpv.CuentaCentral
LinDiario.FechaRegistro=TablaCaja.FechaInicio
LinDiario.TipoDocumento=' '
LinDiario.NoDocumento=NumDoc
LinDiario.Descripcion='Trapaso del Tpv'+TablaCaja.CodTpv+'turno'+
                      TablaCaja.NoTurno+' a la central tienda';
LinDiario.CodigoOrigen='DIACOBROS'
LinDiario.Importe= (Importe+Descuadre)
LinDiario.CodTienda=TablaCaja.CodTienda
LinDiario.CodTpv=TablaCaja.CodTpv
LinDiario.TipoMovTraspaso=Trasp.Central
LinDiario.NoTurno=TablaCaja.NoTurno
Modificar registro Lin.Diario

LinDiario se inicializa
LinDiario se filtra NombreDiario sea COBROS
LinDiario se filtra NombreSeccion sea MULTI+TablaCaja.CodTPv
SI encuentra registros LinDiario entonces
     Registrar LinDiario
```

Código Original

```
ConfigTPV.RESET;
ConfigTPV.SETFILTER("Cód. tienda", "Cód. tienda");
ConfigTPV.SETFILTER("Cód. TPV", "Cód. TPV");
ConfigTPV.FIND('-');
ConfigTPV.TESTFIELD("Forma pago EFECTIVO");
ConfigTPV.TESTFIELD("Nº cuenta caja central");
ConfigTPV.TESTFIELD("Cuenta de descuadres");

FormaPago.GET(ConfigTPV."Forma pago EFECTIVO");
FormaPago.TESTFIELD(FormaPago."Bal. Account No.");

IF ImporteTraspaso <= 0 THEN
  ERROR(Text00001);
```

```
MovTraspaso.RESET;
MovTraspaso.SETCURRENTKEY("Tipo movimiento","Cód. tienda","Cód. TPV","Fecha inicio día");
MovTraspaso.SETRANGE("Tipo movimiento", MovTraspaso."Tipo movimiento"::"Trasp. central");
MovTraspaso.SETRANGE("Cód. tienda", "Cód. tienda");
MovTraspaso.SETRANGE("Cód. TPV", "Cód. TPV");
MovTraspaso.SETRANGE("Fecha inicio día", "Fecha inicio día");
IF MovTraspaso.FIND('+') THEN
  NumDoc := INCSTR(MovTraspaso."Nº documento")
ELSE
  NumDoc := "Cód. tienda" + '/' + "Cód. TPV" + '/' +
          FORMAT("Fecha inicio día",0,'<Year,2><Month,2><Day,2>') + '/' + '001';


//COMPROBAMOS LOS IMPORTES DE LAS FORMAS DE PAGO
LinDiaGen.RESET;
LinDiaGen.SETRANGE("Journal Template Name", 'COBROS');
LinDiaGen.SETRANGE("Journal Batch Name", 'MULTI'+ "Cód. TPV");
IF LinDiaGen.FIND('+') THEN
  LinDiaGen.DELETEALL;

LinDiaGen.INIT;
LinDiaGen."Journal Template Name" := 'COBROS';
LinDiaGen."Journal Batch Name" := 'MULTI'+ "Cód. TPV";
LinDiaGen."Line No." := 10000;
LinDiaGen.INSERT;
LinDiaGen."Account Type" := LinDiaGen."Account Type"::"G/L Account";
LinDiaGen.VALIDATE("Account Type");
LinDiaGen.VALIDATE("Account No.", FormaPago."Bal. Account No.");
LinDiaGen."Posting Date" := "Fecha inicio día";
LinDiaGen."Document Type" := LinDiaGen."Document Type"::" ";
LinDiaGen.VALIDATE("Document Type");
LinDiaGen."Document No." := NumDoc;
LinDiaGen.Description := 'Traspaso del tpv ' + "Cód. TPV" + ' turno ' +
                    FORMAT("Nº turno") + ' a  la central tienda.';
LinDiaGen."Source Code" := 'DIACOBROS';
LinDiaGen.VALIDATE(Amount ,-ImporteTraspaso);
LinDiaGen."Cód. tienda" := "Cód. tienda";
LinDiaGen."Cód. TPV" := "Cód. TPV";
LinDiaGen."Tipo mov. traspaso" := LinDiaGen."Tipo mov. traspaso"::"Trasp. central";
LinDiaGen."Nº turno" := "Nº turno";
LinDiaGen.MODIFY;

IF Descuadre <> 0 THEN
BEGIN
  LinDiaGen.INIT;
  LinDiaGen."Journal Template Name" := 'COBROS';
  LinDiaGen."Journal Batch Name" := 'MULTI'+ "Cód. TPV";
  LinDiaGen."Line No." := 20000;
  LinDiaGen.INSERT;
  LinDiaGen."Account Type" := LinDiaGen."Account Type"::"G/L Account";
  LinDiaGen.VALIDATE("Account Type");
  LinDiaGen.VALIDATE("Account No.", ConfigTPV."Cuenta de descuadres");
```

```
  LinDiaGen."Posting Date" := "Fecha inicio día";
  LinDiaGen."Document Type" := LinDiaGen."Document Type"::" ";
  LinDiaGen.VALIDATE("Document Type");
  LinDiaGen."Document No." := NumDoc;
  LinDiaGen.Description := 'Traspaso del tpv ' + "Cód. TPV" + ' turno ' +
                    FORMAT("Nº turno") + ' a  la central tienda.';
  LinDiaGen."Source Code" := 'DIACOBROS';
  LinDiaGen.VALIDATE(Amount ,-Descuadre);
  LinDiaGen."Cód. tienda" := "Cód. tienda";
  LinDiaGen."Cód. TPV" := "Cód. TPV";
  LinDiaGen."Tipo mov. traspaso" := LinDiaGen."Tipo mov. traspaso"::"Trasp. central";
  LinDiaGen."Nº turno" := "Nº turno";
  LinDiaGen.MODIFY;
END;
LinDiaGen.INIT;
LinDiaGen."Journal Template Name" := 'COBROS';
LinDiaGen."Journal Batch Name" := 'MULTI'+ "Cód. TPV";
LinDiaGen."Line No." := 30000;
LinDiaGen.INSERT;
LinDiaGen."Account Type" := LinDiaGen."Account Type"::"G/L Account";
LinDiaGen.VALIDATE("Account Type");
LinDiaGen.VALIDATE("Account No.", ConfigTPV."Nº cuenta caja central");

LinDiaGen."Posting Date" := "Fecha inicio día";
LinDiaGen."Document Type" := LinDiaGen."Document Type"::" ";
LinDiaGen.VALIDATE("Document Type");
LinDiaGen."Document No." := NumDoc;
LinDiaGen.Description := 'Traspaso del tpv ' + "Cód. TPV" + ' turno ' +
                    FORMAT("Nº turno") + ' a  la central tienda.';
LinDiaGen."Source Code" := 'DIACOBROS';
LinDiaGen.VALIDATE(Amount ,ImporteTraspaso+Descuadre);
LinDiaGen."Cód. tienda" := "Cód. tienda";
LinDiaGen."Cód. TPV" := "Cód. TPV";
LinDiaGen."Tipo mov. traspaso" := LinDiaGen."Tipo mov. traspaso"::"Trasp. central";
LinDiaGen."Nº turno" := "Nº turno";
LinDiaGen.MODIFY;
//REGISTRAMOS EL DIARIO
LinDiaGen.RESET;
LinDiaGen.SETRANGE("Journal Template Name", 'COBROS');
LinDiaGen.SETRANGE("Journal Batch Name", 'MULTI'+ "Cód. TPV");
IF LinDiaGen.FIND('-') THEN;
    CODEUNIT.RUN(CODEUNIT::"Gen. Jnl.-Post",LinDiaGen);
"Error Envío":= TRUE;
MODIFY;
COMMIT;
```

EnviarCaja

PseudoCódigo

ConfigTpv se inicializa
ConfigTpv se filtra CodTienda sea TablaCaja.CodTienda
ConfigTpv se filtra CodTpv sea TablaCaja.CodTpv
ConfigTpv se posiciona primer registro
Si ConfigTpv.ModoTrabajo=On-Line entonces

```
   Si ServicioWeb.ST_AbrirLogin(ConfigTpv.CodTienda,ConfigTpv.Password,
                                        Verdadero) entonces
      Si ServicioWeb.TPV_Cierre_Caja2(ConfigTpv.CodTienda,ConfigTpv.Password,
                                                TablaCaja) entonces
         TablaCaja.ErrorEnvio=Falso
         Modificar registro TablaCaja
      Sino
         ConfigTpv.NoFallos=ConfigTpv.NoFallos.+1
         Modificar registro ConfigTpv
        Si ConfigTpv.NoFallos>3 entonces
           ConfigTpv.FechaOffLine=Hoy
           ConfigTpv.HoraOffLine=Hoy
           ConfigTpv.ModoTrabajo=Off-Line
           Modificar registro ConfigTpv
Sino
   ConfigTpv.NoFallos=ConfigTpv.NoFallos.+1
   Modificar registro ConfigTpv
   Si ConfigTpv.NoFallos>3 entonces
      ConfigTpv.FechaOffLine=Hoy
      ConfigTpv.HoraOffLine=Hoy
      ConfigTpv.ModoTrabajo=Off-Line
       Modificar registro ConfigTpv
```

Código Original

```
ConfigTPV.RESET;
ConfigTPV.SETFILTER("Cód. tienda", "Cód. tienda");
ConfigTPV.SETFILTER("Cód. TPV", "Cód. TPV");
ConfigTPV.FIND('-');
IF ConfigTPV."Modo de Trabajo"= ConfigTPV."Modo de Trabajo"::"On-Line" THEN
BEGIN
 IF ws.ST_Abrir_Login(ConfigTPV."Cód. tienda",ConfigTPV.Password,TRUE) THEN
 BEGIN
   IF ws.TPV_Cierre_Caja2(ConfigTPV."Cód. tienda",ConfigTPV.Password,Rec) THEN
   BEGIN
     "Error Envío":= FALSE;
     MODIFY;
   END
   ELSE
   BEGIN
     ConfigTPV."Nº de fallos":=ConfigTPV."Nº de fallos"+1;
     ConfigTPV.MODIFY;
     IF ConfigTPV."Nº de fallos" > 3 THEN
     BEGIN
       ConfigTPV."Fecha entrada Off-Line":=WORKDATE;
       ConfigTPV."Hora Entrada Off-Line":=TIME;
       ConfigTPV."Modo de Trabajo":=ConfigTPV."Modo de Trabajo"::"Off-Line";
       ConfigTPV.MODIFY;
     END;
   END;
 END
 ELSE
 BEGIN
   ConfigTPV."Nº de fallos":=ConfigTPV."Nº de fallos"+1;
   ConfigTPV.MODIFY;
```

```
    IF ConfigTPV."Nº de fallos" > 3 THEN
    BEGIN
      ConfigTPV."Fecha entrada Off-Line":=WORKDATE;
      ConfigTPV."Hora Entrada Off-Line":=TIME;
      ConfigTPV."Modo de Trabajo":=ConfigTPV."Modo de Trabajo"::"Off-Line";
      ConfigTPV.MODIFY;
    END;
  END;
END;
```

Gasto Tienda

En este apartado, si en algún momento la tienda debe coger dinero de la caja para cualquier gasto inesperado, es decir, productos de limpieza, productos del ordenador, etc. Ese gasto debe constar en la caja del día. Entonces, el responsable de tienda debe ir a la pantalla de gasto, rellenar una serie de datos y registrarlo para que a la hora de realizar el cierre se tenga en cuenta.

Ilustración 60 Pantalla para registrar un gasto externo que haya tenido la tienda

Registrar Gasto

PseudoCódigo

```
Si No Confirma('Está seguro de realizar el gasto de tienda por %1?',Falso,Importe)
   Salir
ConfigTpv se inicializa
ConfigTpv se filtra CodTienda sea TablaTiendas.CodTienda
ConfigTpv se filtra CodTpv sea Numero
ConfigTpv se posiciona primer registro

TablaCaja se inicializa
TablaCaja se filtra CodTienda sea ConfigTpv.CodTienda
TablaCaja se filtra CodTpv sea ConfigTpv.CodTpv
TablaCaja se filtra FechaInicio sea Hoy
TablaCaja se filtra Activa sea Verdadero
Si no encuentra registros TablaCaja entonces
   ERROR('No se encuentra la caja activa de: \ tienda -> %1\ $ fecha inicio -> %3.',
            Tiendas.CodTienda,ConfigTienda.CodTpv,Hoy)
Si NumDoc='' entonces
    ERROR('Ha de indicar un nº documento o de ticket para registrar el gasto.')
Si TipoGasto='' entonces
    ERROR('Ha de introducir el tipo de gasto.')
Si DescGasto='' entonces
    ERROR('Introduzca la descripción del gasto.');
Si CuentaTraspaso='' entonces
    ERROR('Ha de indicar una cuenta destino.');
Si CuentaOrigen='' entonces
    ERROR('Ha de indicar una cuenta origen.');
Si FechaTraspaso=0D entonces
    ERROR('Ha de indicar una fecha del gasto.');
Si Importe=0 entonces
    ERROR('Ha de indicar un importe del gasto.')
TablaCaja.Calcula(TotalGastosTpv,TotalTraspaso,VentasEfectivo,ReservasEfectivo,
                     AbonosEfectivo,ReservasCanceladasEfectivo)
ImporteMaximo=TablaCaja.SaldoInicial+TablaCaja.VentasEfectivo+
                     TablaCaja.ReservasEfectivo-TablaCaja.AbonosEfectivo-
                     TablaCaja.ReservasCanceladasEfectivo-TablaCaja.SaldoFinal
                     -TablaCaja.TotalTraspaso-TablaCaja.TotalGastos
Si Importe>ImporteMaximo entonces
    ERROR('Solo puede gastar %1.',ImporteMaximo)
RegistroTrans(Importe)
Si ConfigTpv.ModoTrabajo=On-Line
    Si ServicioWeb.ST_Abrir_Login(ConfigTpv.CodTienda,ConfigTpv.Password)
        MovTraspaso se inicializa
        MovTraspaso se filtra TipoMovimiento sea Gasto tienda
        MovTraspaso se filtra NoDocumento sea NumDoc
        MovTraspaso se filtra FechaInicio sea FechaTraspaso
        Si no encuentra registros MovTraspaso entonces
           ERROR('ERROR Grave, no existe histórico de gasto.')
        Si No ServicioWeb.TPV_RegDiario(ConfigTpv.CodTienda,ConfigTpv:Password
              ,FechaTraspaso,0,NumDoc,0,CuentaOrigen,DesGasto,Importe,0,
              CuentaTraspaso,ConfigTpv.CodTpv,TablaCaja.NoTurno,3,'-') entonces
           MovTraspaso se inicializa
           MovTraspaso se filtra TipoMovimiento sea Gasto tienda
```

```
            MovTraspaso se filtra NoDocumento sea NumDoc
            MovTraspaso se filtra FechaInicio sea FechaTraspaso
            Si no encuentra registros MovTraspaso entonces
               ERROR('ERROR Grave, no existe histórico de gasto.')
            Sino
               MovTraspaso.ErrorEnvio=Verdadero
               Modificar registro MovTraspaso
               ConfigTpv.NoFallos=Config.NoFallos+1
               Modificar registro ConfigTpv
               Si ConfigTpv.NoFallos>3 entonces
                  ConfigTpv.FechaOffLine=Hoy
                  ConfigTpv.HoraOffLine=Hoy
                  ConfigTpv.ModoTrabajo=Off-Line
                  Modificar registro ConfigTpv
Sino
   MovTraspaso se inicializa
   MovTraspaso se filtra TipoMovimiento sea Gasto tienda
   MovTraspaso se filtra NoDocumento sea NumDoc
   MovTraspaso se filtra FechaInicio sea FechaTraspaso
   Si no encuentra registros MovTraspaso entonces
      ERROR('ERROR Grave, no existe histórico de gasto.')
    Sino
      MovTraspaso.ErrorEnvio=Verdadero
       Modificar registro MovTraspaso
       ConfigTpv.NoFallos=Config.NoFallos+1
        Modificar registro ConfigTpv
        Si ConfigTpv.NoFallos>3 entonces
           ConfigTpv.FechaOffLine=Hoy
            ConfigTpv.HoraOffLine=Hoy
            ConfigTpv.ModoTrabajo=Off-Line
             Modificar registro ConfigTpv
Sino
  MovTraspaso se inicializa
  MovTraspaso se filtra TipoMovimiento sea Gasto tienda
  MovTraspaso se filtra NoDocumento sea NumDoc
  MovTraspaso se filtra FechaInicio sea FechaTraspaso
 Si no encuentra registros MovTraspaso entonces
    ERROR('ERROR Grave, no existe histórico de gasto.')
 Sino
    MovTraspaso.ErrorEnvio=Verdadero
    Modificar registro MovTraspaso
    ConfigTpv.NoFallos=Config.NoFallos+1
    Modificar registro ConfigTpv
    Si ConfigTpv.NoFallos>3 entonces
       ConfigTpv.FechaOffLine=Hoy
       ConfigTpv.HoraOffLine=Hoy
       ConfigTpv.ModoTrabajo=Off-Line
       Modificar registro ConfigTpv
Importe=0
CuentaTraspaso=0
CodBanco=''
MovTraspaso se inicializa
MovTraspaso se filtra TipoMovimiento sea Gasto tienda
MovTraspaso se filtra NoDocumento sea NumDoc
```

```
MovTraspaso se filtra FechaInicio sea FechaTraspaso
Si no encuentra registros MovTraspaso entonces
  ERROR(‘ERROR Grave, no existe histórico de gasto.’)
Case ConfigTpv.Impresora
      Negra:
            InformeGastosN.Filtrar(MovTraspaso)
            InformeGastoN.UsarOpciones(Falso)
            InformeGastosN.Ejecutar
      Negra:
            InformeGastosB.Filtrar(MovTraspaso)
            InformeGastoB.UsarOpciones(Falso)
            InformeGastosB.Ejecutar
Mensaje(‘Gasto Registrado’);
```

Código Original

```
IF NOT CONFIRM(Text00001,FALSE,ImpATraspasar) THEN
  EXIT;

ConfigTPV.RESET;
ConfigTPV.SETRANGE("Cód. tienda", "Cód. tienda");
ConfigTPV.SETFILTER("Cód. TPV",NTPV);
ConfigTPV.FIND('-');

Caja.RESET;
Caja.SETRANGE("Cód. tienda",ConfigTPV."Cód. tienda");
Caja.SETRANGE("Cód. TPV",ConfigTPV."Cód. TPV");
Caja.SETRANGE("Fecha inicio día",WORKDATE);
Caja.SETFILTER(Activa,'Sí');
IF NOT Caja.FIND('-') THEN
  ERROR(Text00002+
      Text00003+
      Text00004+
      Text00005,"Cód. tienda",ConfigTPV."Cód. TPV",WORKDATE);

IF NumDoc = '' THEN
  ERROR(Text00006);

IF TipoGasto = TipoGasto::" " THEN
  ERROR(Text00007);

IF DescGasto = '' THEN
  ERROR(Text00008);

IF CuentaTraspaso = '' THEN
  ERROR(Text00009);

IF CuentaOrigen = '' THEN
  ERROR(Text00010);

IF FechaTraspaso = 0D THEN
  ERROR(Text00011);

IF ImpATraspasar = 0 THEN
  ERROR(Text00012);
```

```
Caja.CALCFIELDS("Total gastos tpv","Total traspasado a central");
Caja.CALCFIELDS("Ventas Efectivo","Reservas Efectivo","Abonos Efectivo","Reservas
Cancelada Efectivo");

ImpMaximo := Caja."Saldo inicial" + Caja."Ventas Efectivo"  + Caja."Reservas Efectivo" -
            Caja."Abonos Efectivo" - Caja."Reservas Cancelada Efectivo" - Caja."Saldo final
TPV" -
            Caja."Total traspasado a central" - Caja."Total gastos tpv";

IF ImpATraspasar > ImpMaximo THEN
  ERROR(Text00013,ImpMaximo);

RegTraspaso(ImpATraspasar);

IF ConfigTPV."Modo de Trabajo"= ConfigTPV."Modo de Trabajo"::"On-Line" THEN
  BEGIN
    IF ws.ST_Abrir_Login(ConfigTPV."Cód. tienda",ConfigTPV.Password,TRUE) THEN
      BEGIN

        MovTraspaso.RESET;
        MovTraspaso.SETCURRENTKEY("Tipo movimiento", "Nº documento", "Fecha inicio
día");
        MovTraspaso.SETRANGE("Tipo movimiento", MovTraspaso."Tipo
movimiento"::"Gasto tienda");
        MovTraspaso.SETRANGE("Nº documento", NumDoc);
        MovTraspaso.SETRANGE("Fecha inicio día", FechaTraspaso);
        IF NOT MovTraspaso.FIND('+') THEN
            ERROR(Text00014);

        IF NOT ws.TPV_RegDiario(ConfigTPV."Cód.
tienda",ConfigTPV.Password,FechaTraspaso,
0,NumDoc,0,CuentaOrigen,DescGasto,ImpATraspasar,0,CuentaTraspaso,
ConfigTPV."Cód. tienda",ConfigTPV."Cód. TPV",Caja."Nº turno",3,'-') THEN
          BEGIN
            MovTraspaso.RESET;
            MovTraspaso.SETCURRENTKEY("Tipo movimiento", "Nº documento", "Fecha
inicio día");
            MovTraspaso.SETRANGE("Tipo movimiento", MovTraspaso."Tipo
movimiento"::"Gasto tienda");
            MovTraspaso.SETRANGE("Nº documento", NumDoc);
            MovTraspaso.SETRANGE("Fecha inicio día", FechaTraspaso);
            IF NOT MovTraspaso.FIND('+') THEN
              ERROR(Text00014)
            ELSE BEGIN
                MovTraspaso."Error envío":= TRUE;
                MovTraspaso.MODIFY;
                ConfigTPV."Nº de fallos":=ConfigTPV."Nº de fallos"+1;
                ConfigTPV.MODIFY;
                IF ConfigTPV."Nº de fallos" > 3 THEN
                  BEGIN
                    ConfigTPV."Fecha entrada Off-Line":=WORKDATE;
                    ConfigTPV."Hora Entrada Off-Line":=TIME;
                    ConfigTPV."Modo de Trabajo":=ConfigTPV."Modo de Trabajo"::"Off-Line";
```

```
                ConfigTPV.MODIFY;
              END;
            END;
          END;
    END ELSE
      BEGIN
        MovTraspaso.RESET;
        MovTraspaso.SETCURRENTKEY("Tipo movimiento", "Nº documento", "Fecha inicio
día");
        MovTraspaso.SETRANGE("Tipo movimiento", MovTraspaso."Tipo
movimiento"::"Gasto tienda");
        MovTraspaso.SETRANGE("Nº documento", NumDoc);
        MovTraspaso.SETRANGE("Fecha inicio día", FechaTraspaso);
        IF NOT MovTraspaso.FIND('+') THEN
          ERROR(Text00014)
        ELSE BEGIN
          MovTraspaso."Error envío":= TRUE;
          MovTraspaso.MODIFY;
          ConfigTPV."Nº de fallos":=ConfigTPV."Nº de fallos"+1;
          ConfigTPV.MODIFY;
          IF ConfigTPV."Nº de fallos" > 3 THEN
            BEGIN
              ConfigTPV."Fecha entrada Off-Line":=WORKDATE;
              ConfigTPV."Hora Entrada Off-Line":=TIME;
              ConfigTPV."Modo de Trabajo":=ConfigTPV."Modo de Trabajo"::"Off-Line";
              ConfigTPV.MODIFY;
            END;
        END;

      END;
END
ELSE
BEGIN
  MovTraspaso.RESET;
  MovTraspaso.SETCURRENTKEY("Tipo movimiento", "Nº documento", "Fecha inicio día");
  MovTraspaso.SETRANGE("Tipo movimiento", MovTraspaso."Tipo movimiento"::"Gasto
tienda");
  MovTraspaso.SETRANGE("Nº documento", NumDoc);
  MovTraspaso.SETRANGE("Fecha inicio día", FechaTraspaso);
  IF NOT MovTraspaso.FIND('+') THEN
    ERROR(Text00014)
  ELSE BEGIN
    MovTraspaso."Error envío":= TRUE;
    MovTraspaso.MODIFY;
    ConfigTPV."Nº de fallos":=ConfigTPV."Nº de fallos"+1;
    ConfigTPV.MODIFY;
    IF ConfigTPV."Nº de fallos" > 3 THEN BEGIN
      ConfigTPV."Fecha entrada Off-Line":=WORKDATE;
      ConfigTPV."Hora Entrada Off-Line":=TIME;
      ConfigTPV."Modo de Trabajo":=ConfigTPV."Modo de Trabajo"::"Off-Line";
      ConfigTPV.MODIFY;
    END;
  END;
END;
```

```
ImpATraspasar := 0;
CuentaTraspaso := '';
CodBanco := '';

MovTraspaso.RESET;
MovTraspaso.SETCURRENTKEY("Tipo movimiento", "Nº documento", "Fecha inicio día");
MovTraspaso.SETRANGE("Tipo movimiento", MovTraspaso."Tipo movimiento"::"Gasto
tienda");
MovTraspaso.SETRANGE("Nº documento", NumDoc);
MovTraspaso.SETRANGE("Fecha inicio día", FechaTraspaso);
IF NOT MovTraspaso.FIND('+') THEN
  ERROR(Text00014);

CASE ConfigTPV."Impresora Tickets" OF
    ConfigTPV."Impresora Tickets"::Negra:
      BEGIN
        CLEAR(rptGastosNegra);
        rptGastosNegra.SETTABLEVIEW(MovTraspaso);
        rptGastosNegra.USEREQUESTFORM(FALSE);
        rptGastosNegra.RUNMODAL;
      END;
    ConfigTPV."Impresora Tickets"::Blanca:
      BEGIN
        CLEAR(rptGastosBlanca);
        rptGastosBlanca.SETTABLEVIEW(MovTraspaso);
        rptGastosBlanca.USEREQUESTFORM(FALSE);
        rptGastosBlanca.RUNMODAL;
      END;
END;
MESSAGE(Text00015);
CurrForm.CLOSE;
```

RegistroTrans(Importe)

PseudoCódigo

```
Si ImporteTraspaso<=0 entonces
    ERROR('Ha de introducir un importe a gastar.');

LinDiario se inicializa
LinDiario se filtra NombreDiario sea COBROS
LinDiario se filtra NombreSeccion sea MULTI+Numero
Si encuentra registros LinDiario entonces
   Borrar Todos los registros LinDiario
LinDiario se inicializa
LinDiario.NombreDiario='COBROS'
LinDiario.NombreSeccion='MULTI'+Numero
LinDiario.NoLine=10000
Insertar registro LinDiario
LinDiario.TipoCuenta=Cuenta
LinDiario.NoCuenta=CuentaOrigen
LinDiario.FechaRegistro=FechaTraspaso
LinDiario.TipoDocumento=''
LinDiario.NoDocumento=NumDoc
LinDiario.Descripcion='Gasto de la tienda'+TablaTiendas.CodTienda
```

LinDiario.CodigoOrigen=DIACOBROS
LinDiario.TipoCuentaContra=Cuenta
LinDiario.NoCuentaContra=CuentaTraspaso
LinDiario.Importe= -(Importe)
LinDiario.CodTienda=Tiendas.CodTienda
LinDiario.CodTpv=Config.CodTpv
LinDiario.TipoMovTraspaso=Gasto Tienda
LinDiario.NoTurno=TablaCaja.NoTurno
LinDiario.Descripcion=DescGasto
Modificar registro LinDiario
LinDiario se inicializa
LinDiario se filtra NombreDiario sea COBROS
LinDiario se filtra NombreSeccion sea MULTI+Numero
Si encuentra registros LinDiario entonces
   Registrar Diario

Código Original

```
IF ImporteTraspaso <= 0 THEN
  ERROR(Text00001);

LinDiaGen.RESET;
LinDiaGen.SETRANGE("Journal Template Name", 'COBROS');
LinDiaGen.SETRANGE("Journal Batch Name", 'MULTI'+ NTPV);
IF LinDiaGen.FIND('+') THEN
  LinDiaGen.DELETEALL;
LinDiaGen.INIT;
LinDiaGen."Journal Template Name" := 'COBROS';
LinDiaGen."Journal Batch Name" := 'MULTI'+ NTPV;
LinDiaGen."Line No." := 10000;
LinDiaGen.INSERT;
LinDiaGen."Account Type" := LinDiaGen."Account Type"::"G/L Account";
LinDiaGen.VALIDATE("Account Type");
LinDiaGen.VALIDATE("Account No.", CuentaOrigen);
LinDiaGen."Posting Date" := FechaTraspaso;
LinDiaGen."Document Type" := LinDiaGen."Document Type"::" ";
LinDiaGen.VALIDATE("Document Type");
LinDiaGen."Document No." := NumDoc;
LinDiaGen.Description := 'Gasto de la tienda ' + "Cód. tienda";
LinDiaGen."Source Code" := 'DIACOBROS';
LinDiaGen."Bal. Account Type" := LinDiaGen."Bal. Account Type"::"G/L Account";
LinDiaGen."Bal. Account No." :=  CuentaTraspaso;
LinDiaGen.VALIDATE(Amount ,-ImporteTraspaso);
LinDiaGen."Cód. tienda" := "Cód. tienda";
LinDiaGen."Cód. TPV" := ConfigTPV."Cód. TPV";
LinDiaGen."Tipo mov. traspaso" := LinDiaGen."Tipo mov. traspaso"::"Gasto tienda";
LinDiaGen."Nº turno" := Caja."Nº turno";
LinDiaGen.Description := DescGasto;
LinDiaGen.MODIFY;
LinDiaGen.RESET;
LinDiaGen.SETRANGE("Journal Template Name", 'COBROS');
LinDiaGen.SETRANGE("Journal Batch Name", 'MULTI'+ NTPV);
IF LinDiaGen.FIND('-') THEN;
   CODEUNIT.RUN(CODEUNIT::"Gen. Jnl.-Post",LinDiaGen);
```

Ticket Venta

Esta funcionalidad se refiere a realizar una venta a un cliente. Consta de dos pantallas.

En la primera, se configura que productos y a que cliente se quiere realizar la venta. Por defecto, la venta se realiza a un cliente genérico que tiene el programa, ya que no es obligatorio que se haga a un cliente en concreto, a menos que el propio cliente pida que se haga la factura a su nombre. Para ese caso, se debe dar de alta como cliente en la base de datos y cambiar el cliente en la cabecera de la venta (ilustración 61).

Por otro lado, se insertan los productos que va a comprar el cliente. La inserción se puede realizar de diferentes maneras: Por su código interno, por su código de barras o por la referencia de proveedor. Para cada opción, el sistema realiza una búsqueda diferente, que se explica más detalladamente en el código.

Una vez se hayan configurado todos los datos requeridos, se debe pasar a otra pantalla mediante F11. Antes de pasar a la otra pantalla, el sistema comprueba si alguno de los productos insertados se encuentra en alguna promoción para recalcular su precio de venta.

En la siguiente pantalla (ilustración 62), es donde se indica que vendedor hace la venta (-a final de mes, RRHH calcula las comisiones de cada uno) y la forma de pago que va a utilizar el cliente para pagar. Después de configurar los datos, el vendedor se dispone a facturar la venta, para que el sistema imprima el ticket que se entrega al cliente y así dar por finalizada la venta.

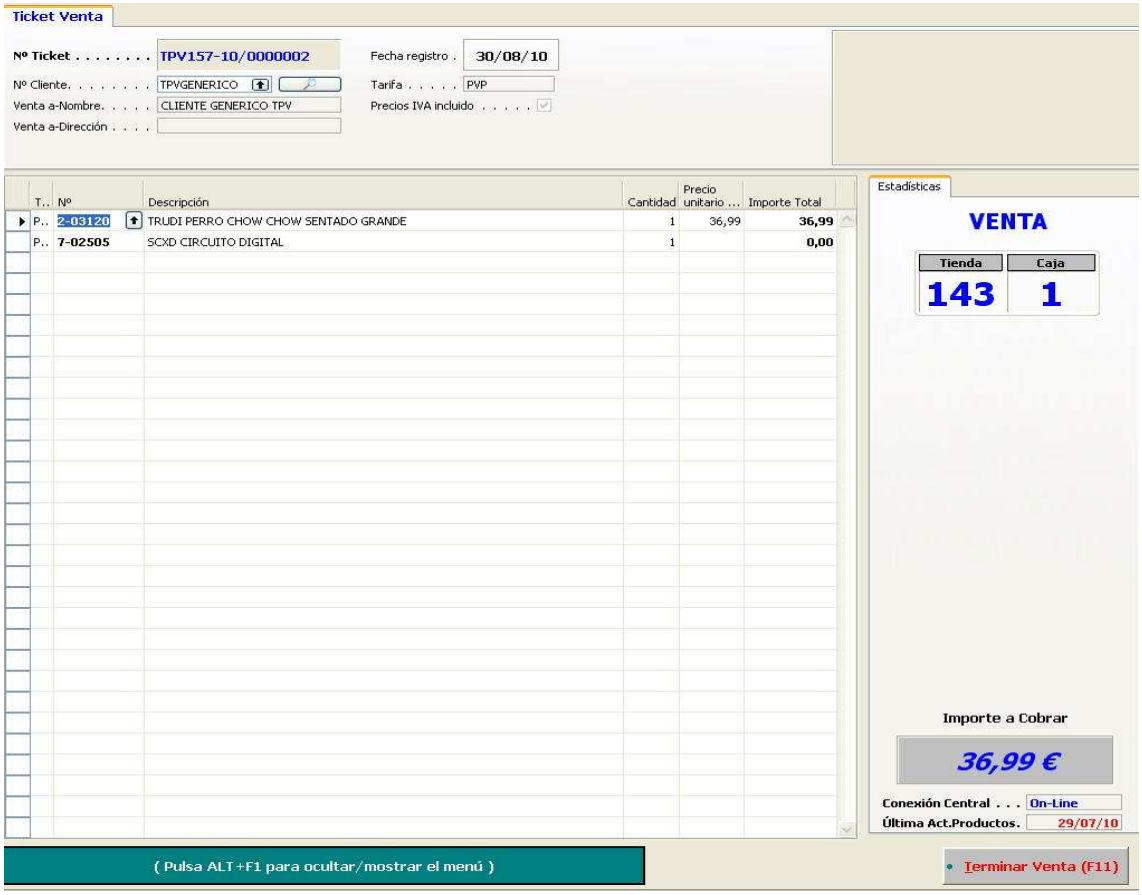


Ilustración 61 Pantalla donde se indica que productos y a que cliente se le realiza la venta

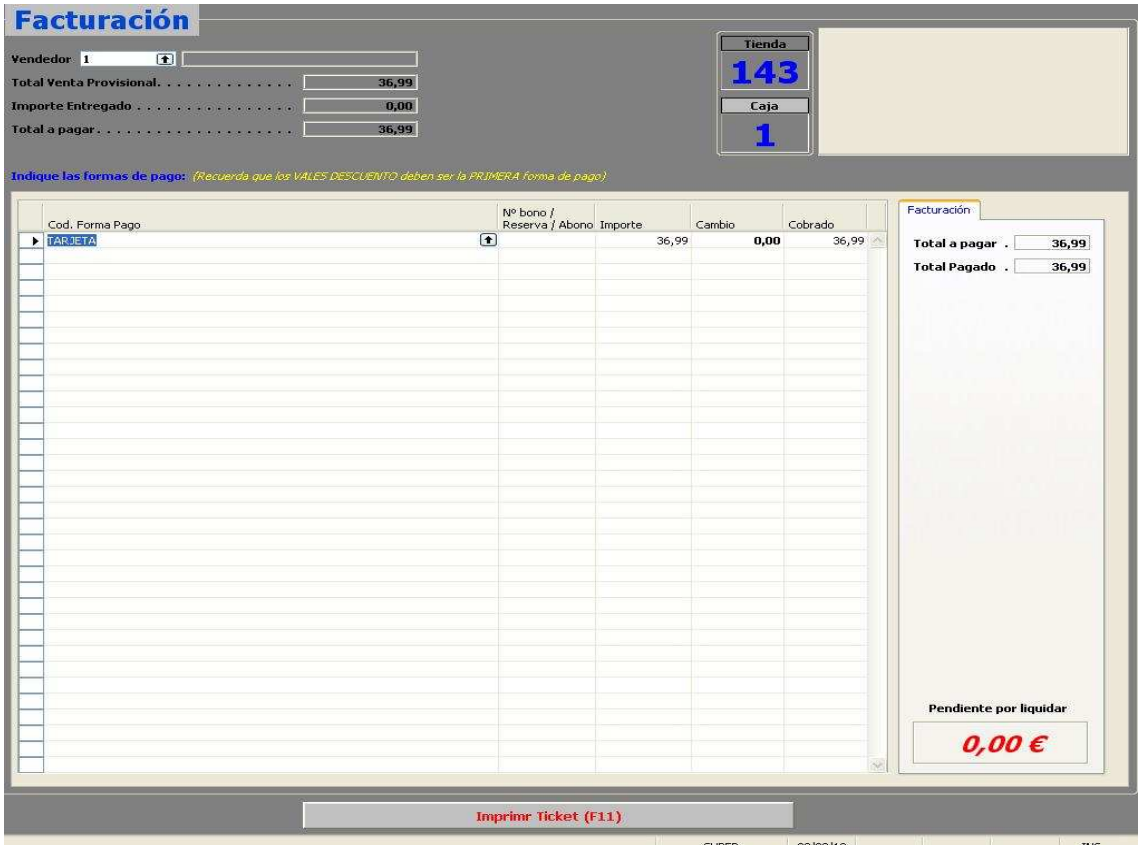


Ilustración 62 Pantalla donde se indica la forma de pago y el vendedor que realiza la venta

Abrir Pantalla(CabVenta)

PseudoCódigo

```
ConfigTpv se filtra CodTpv sea Numero
ConfigTpv se posiciona primer registro
CodTienda=ConfigTpv.CodTienda
FormaPagoEfec=ConfigTpv.FormaPagoEfectivo
TablaFormaPago.Coge(Codigo,MULTI+Numero)
Si encuentra registros TablaFormaPago entonces
   Borrar registro
TablaCaja se filtra CodTpv sea ConfigTpv.CodTpv
TablaCaja se filtra FechaInicio sea Hoy
TablaCaja se filtra Activa sea Verdadero
Si no encuentra registros entonces
    ERROR('Recuerde hacer el inicio de día en TPV %1 y fecha %2.',Numero, HOY)
Sino Si No TablaCaja.Apertura entonces
    ERROR('Ha de registrar el inicio de día en tpv %1 y fecha %2.,Numero,HOY)
CabVenta se filtra CodTpv sea Numero
CabVenta se posiciona primer registro
Si CabVenta.FechRegistro<>Hoy Y CabVenta.FechaRegistro<>0D entonces
    CabVenta.FechaRegistro=Hoy
    CabVenta.FechaPedido=Hoy
    CabVenta.FechaEnvio=Hoy
    Modificar registro CabVenta
```

Código Original

```
ConfTPV.SETFILTER("Cód. TPV",NTpv);
ConfTPV.FINDFIRST();
CodTienda := ConfTPV."Cód. tienda";
ConfTPV.TESTFIELD("Forma pago EFECTIVO");
FormPagoEfec := ConfTPV."Forma pago EFECTIVO";

Forma.SETRANGE(Code,'MULTI'+ NTpv);
IF Forma.FINDFIRST() THEN
  Forma.DELETE;
Inicio.SETRANGE("Cód. TPV",ConfTPV."Cód. TPV");
Inicio.SETRANGE(Inicio."Fecha inicio día",WORKDATE);
Inicio.SETRANGE(Inicio.Activa, TRUE);
IF NOT Inicio.FINDLAST() THEN
   ERROR (Text0001,NTpv,WORKDATE)
ELSE IF NOT Inicio.Apertura THEN
   ERROR(Text0002,NTpv,WORKDATE);

SETFILTER("Cód. TPV",NTpv);
IF Rec.FINDFIRST() THEN;

IF ("Posting Date" <> WORKDATE) AND("Posting Date" <> 0D) THEN BEGIN
  VALIDATE("Posting Date",WORKDATE);
  VALIDATE("Order Date",WORKDATE);
  VALIDATE("Shipment Date",WORKDATE);
   MODIFY;
END;
```

BuscarProduco(LinVenta)

PseudoCódigo

```
Si No TablaProducto.Coge(LinVenta.No) entonces
   TablaCodigoBarras se inicializa
   TablaCodigoBarras se filtra CodigoBarras sea LinVenta.No
   Si encuentra registros TablaCodigoBarras entonces
      TablaProducto.Coge(TablaCodigoBarras.Producto)
      LinVenta.No=TablaProducto.No
   Sino
      TablaProducto se inicializa
      TablaProducto se filtra CodProveedor sea LinVenta.No
      Si no encuentra registros TablaProducto entonces
          ERROR(‘No se ha encontrado producto %1’,LinVenta.No)
       i=0
       Repetir
           i=i+1;
           MatrizProv[i,1]=Formateo(TablaProducto.No)
           MatrizProv[i,2]=TablaProducto.CodProv
           Si i=1 entonces
              cvProds=MatrizProv[i,1]+’ – ‘+TablaProducto.Descripcion
          Sino
              Si ((Longitud(cvProds)+Longitud(MatrizProv[i,1]+’ – ‘+
                  TablaProducto.Descripcion))>MaxLongitud(cvProds)) entonces
                  ERROR(‘Existen demasiados productos’)
              cvProds=cvProds+’,’+ MatrizProv[i,1]+’ – ‘+TablaProducto.Descripcion
       Hasta registro TablaProducto sea vacío
       Si i>1 entonces
          nvLin=STRMENU(cvProds)
          Si nvLin=0 entonces
             ERROR(‘Cancelado’)
          LinVenta.No=MatrizProv[nvLin,1]
       Sino
          LinVenta.No=MatrizProv[1,1]
Sino
  LinVenta.No=TablaProducto.No
```

Código Original

```
 IF NOT Item.GET("No.") THEN
 BEGIN
   CBarrasAsociado.RESET;
   CBarrasAsociado.SETCURRENTKEY("Cod. barras asociado");
   CBarrasAsociado.SETRANGE("Cod. barras asociado","No.");
   IF CBarrasAsociado.FIND('-') THEN
   BEGIN
     Item.GET(CBarrasAsociado.Producto);
     "No." := Item."No.";
   END
   ELSE
   BEGIN
     Item.RESET;
     Item.SETCURRENTKEY("Vendor Item No.","Vendor No.");
     Item.SETRANGE("Vendor Item No.","No.");
```

```
    IF NOT Item.FIND('-') THEN
      ERROR(Text00001,"No.");
    i:=0;
    REPEAT
      i+=1;
      aRefProv[i,1]:=FORMAT(Item."No.");
      aRefProv[i,2]:=Item."Vendor Item No.";
      IF i=1 THEN
        cvProds:=aRefProv[i,1] + ' - ' + Item.Description
      ELSE
        BEGIN
          IF ((STRLEN(cvProds) + STRLEN(aRefProv[i,1] + ' - ' + Item.Description)) >
MAXSTRLEN(cvProds)) THEN
            ERROR(Text00002);
          cvProds:=cvProds+ ',' + aRefProv[i,1] + ' - ' + Item.Description;
        END;
    UNTIL Item.NEXT=0;
    IF i > 1 THEN
    BEGIN
      nvLin:=STRMENU(cvProds);
      IF nvLin=0 THEN
        ERROR(Text00003);
      "No.":=aRefProv[nvLin,1];
    END
    ELSE
      "No.":=aRefProv[1,1];
  END;
 END;
```

PreCalculaPromociones(Tipo,CabVenta.No)

PseudoCódigo

```
TablaPromocion se inicializa
TablaPromocion se filtra Emision/Consumo sea Tipo
TablaPromocion se filtra FechaInicio sea menor o igual que Hoy
TablaPromocion se filtra FechaFin sea mayor o igual Hoy O OD
Si encuentra registros TablaPromocion entonces
   Repetir
      Caso TablaPromocion.CodPromocion
         ‘RAPELFLY09’:RapelVenta(TablaPromocion.CodPromocion, CabVenta.No)
         ‘NINCO09’:RapelVenta(TablaPromocion.CodPromocion, CabVenta.No)
         ‘SLOT09’:RapelVenta(TablaPromocion.CodPromocion, CabVenta.No)
         ‘LUNNIS09’:RapelVenta(TablaPromocion.CodPromocion, CabVenta.No)
         ‘TELESTORY’:RapelVenta(TablaPromocion.CodPromocion, CabVenta.No)
         ‘PINTURAS10’:RapelVenta(TablaPromocion.CodPromocion, CabVenta.No)
   Hasta registro TablaPromocion sea vacío
```

Código Original

```
rcdPromocion.RESET;
rcdPromocion.SETRANGE(rcdPromocion."Emision/Consumo",PARDirección);
rcdPromocion.SETFILTER(rcdPromocion."Fecha Inicio",'..%1',WORKDATE);
rcdPromocion.SETFILTER(rcdPromocion."Fecha Fin",'>=%1| %2',WORKDATE,0D);
IF rcdPromocion.FINDSET THEN
BEGIN
```

```
 REPEAT
   CASE rcdPromocion."Cód. Promocion" OF
       'RAPELFLY09':RapelVenta(rcdPromocion."Cód. Promocion",PARFactura);
       'NINCO09':RapelVenta(rcdPromocion."Cód. Promocion",PARFactura);
       'SLOT09':RapelVenta(rcdPromocion."Cód. Promocion",PARFactura);
       'LUNNIS09':RapelVenta(rcdPromocion."Cód. Promocion",PARFactura);
       'TELESTORY':RapelVenta(rcdPromocion."Cód. Promocion",PARFactura);
       'PINTURAS10':RapelVenta(rcdPromocion."Cód. Promocion",PARFactura);
   END;
 UNTIL rcdPromocion.NEXT=0;
```

RapelVenta(CodPromo,CabVenta.No)

PseudoCódigo

```
LinVenta se inicializa
LinVenta se filtra TipoDocumento sea Factura
LinVenta se filtra NoDocumento sea CabVenta.No)
LinVenta se filtra Tipo sea Producto
Si no encuentra registros LinVenta entonces
   Salir
nvProdsenPromo=0
Repetir
   Si ProductoenPromocion(CodPromo,LinVenta.No) entonces
      nvProdsenPromo=nvProdsenPromo+LinVenta.Cantidad
Hasta registro LinVenta sea vacío
Si nvProdsenPromo<=0 entonces
   Salir
nvRestantes=nvProdsenPromo
nvPrecio=0
Repetir
  TablaRapel se inicializa
   TablaRapel se filtra CodPromocion sea CodPromo
   TablaRapel se filtra UnidadesRapel <=nvRestantes
   Si no encuentra registros Tabla Rapel entonces
      ERROR(‘Promoción no Valida)
   nvPrecio=nvPrecio+TablaRapel.PrecioTotal
   nvRestantes=nvRestantes-TablaRapel.Unidades
Hasta nvRestantes=0
nvPrecioU=nvPrecio/nvProdsenPromo
LinVenta se posiciona en el primer registro
Repetir
   Si ProductoenPromocion(CodPromo,LinVenta.No) entonces
       LinVenta.PrecioVenta=nvPrecioU
       Modificar registro LinVenta
Hasta registro LinVenta sea vacío
Message(‘Promocion %1 Activada,CodPromo);
```

Código Original

```
rcdLinVenta.RESET;
rcdLinVenta.SETRANGE("Document Type",rcdLinVenta."Document Type"::Invoice);
rcdLinVenta.SETRANGE("Document No.",PARFac);
rcdLinVenta.SETRANGE(Type,rcdLinVenta.Type::Item);
IF NOT rcdLinVenta.FINDSET() THEN
  EXIT;
```

```
// Calculamos las unidades en promoción
nvProdsEnPromocion:=0;
REPEAT
  IF ProductoEnPromocion(PARPromocion,rcdLinVenta."No.") THEN
     nvProdsEnPromocion+=rcdLinVenta.Quantity;
UNTIL rcdLinVenta.NEXT=0;

IF nvProdsEnPromocion <= 0 THEN
   EXIT;

// En este punto, deberemos calcular el importe unitario de esas unidades
nvRestantes:=nvProdsEnPromocion;
nvPrecioTotal:=0;
REPEAT
  // Buscamos el segmento de Rapel, más óptimo para el cliente
  rcdPromoRapel.RESET;
  rcdPromoRapel.SETRANGE("Cód. Promoción",PARPromocion);
  rcdPromoRapel.SETFILTER("Unidades Rapel", '<= %1',nvRestantes);
  IF NOT rcdPromoRapel.FINDFIRST() THEN
     ERROR(Text001,PARPromocion);
  nvPrecioTotal+=rcdPromoRapel."Precio Total Rapel";
  nvRestantes-=rcdPromoRapel."Unidades Rapel";
UNTIL nvRestantes = 0;
// Volvemos a recorrerlos para actualizar el precio unitario de los productos en promoción
nvPrecioUnitario:=nvPrecioTotal/nvProdsEnPromocion;
rcdLinVenta.FINDSET();
REPEAT
  IF ProductoEnPromocion(PARPromocion,rcdLinVenta."No.") THEN
  BEGIN
     rcdLinVenta.VALIDATE("Unit Price",nvPrecioUnitario);
     rcdLinVenta.MODIFY;
  END;
UNTIL rcdLinVenta.NEXT=0;
MESSAGE('PROMOCION %1 ACTIVADA\\'+
        'Unidades en promoción %2\'+
        'Precio total %3\'+
        'Precio unitario %4'
        ,FORMAT(PARPromocion)
        ,FORMAT(nvProdsEnPromocion),FORMAT(nvPrecioTotal),
         FORMAT(nvPrecioTotal/nvProdsEnPromocion));
```

ProductoenPromocion(CodPromo,Producto)

PseudoCódigo

TablaPromoProd se inicializa
TablaPromoProd se filtra CodPromocion sea CodPromo
TablaPromoProd se filtra CodProducto sea Producto
Salir(TablaPromoProd se posiciona en el primer registro)

Código Original

```
rcdPromoItem.RESET;
rcdPromoItem.SETRANGE("Cód. Promoción",PARPromocion);
rcdPromoItem.SETRANGE("Cód. Producto",PARProducto);
EXIT (rcdPromoItem.FINDFIRST());
```

Facturar(TablaTicket)

PseudoCódigo

```
Si Vendedor='' entonces
    ERROR('Debes indicar un vendedor')
CabVenta se inicializa
CabVenta se filtra TipoDocumento sea Factura
CabVenta se filtra No sea TablaTicket.NoDocumento
CabVenta se posiciona primer registro
Si CabVenta.NoCuentaContra<>'' entonces
    ERROR('Esa cuenta no debe tener contrapartida')
CabVenta.CodigoVendedor=Vendedor
Modificar registro CabVenta
TablaTicket.FechDocumento=Hoy
TablaTicket.HoraDocumento=Tiempo
Modificar registro TablaTicket
FechaTic=TablaTicket.FechaDocumento
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea TablaTicket.CodTpv
ConfigTpv se posiciona primer registro
Si TablaTicket.CodTienda='' Y TablaTicket.CodTpv='' entonces
    ERROR('La venta fue mal generada, suprímala y cree una nueva')
Si TablaTicket.TotalaPagar<>TablaTicket.TotalVentaProv Y CabVenta.Entrega=''
    ERROR('La venta fue mal generada, suprímala y cree una nueva')
Multiforma se inicializa
Multiforma se filtra TipoDocumento sea TablaTicket.TipoDocumento
Multiforma se filtra NoDocumento sea TablaTicket.NoDocumento
Multiforma se filtra TipoPago sea Vale
Si encuentra registros Multiforma entonces
   Repetir
      Si FormaPago.Coge(Multiforma.CodigoForma) Y
         FormaPago.Cumplimentacion=Eleccion de Lista entonces
         TablaBono.Coge(Multiforma.NoBono/Reserva/Abono)
  Hasta registro Multiforma sea vacío
PintarTicket=CabVenta.Vendedor=ConfigTpv.CodClienteTpv
nvImporteDevolver=0
Multiforma se inicializa
Multiforma se filtra TipoDocumento sea TablaTicket.TipoDocumento
Multiforma se filtra NoDocumento sea TablaTicket.NoDocumento
Si encuentra registros Multiforma entonces
    SumaPago=0
    Repetir
       Si Multiforma.TipoPago=Caja entonces
          SumaPago=SumaPago+Multiforma.Cobrado
          nvImporteDevolver=Multiforma.Cambio
       Sino
          SumaPago=SumaPago+Multiforma.Importe
```

```
    Hasta registro Multiforma sea vacío
    Si SumaPago<>(TablaTicket.TotalaPagar-TablaTicket.ImporteEntregado) entonces
       ERROR('Las formas de pago suman %1 y el importe del ticket %2, no cuadran.',
                SumaPago, TablaTicket.TotalaPagar-TablaTicket.ImporteEntregado)
Sino
   ERROR('NO ESTÁN DEFINIDAS LAS FORMAS DE PAGO DEL TICKET.')
Si nvImporteDevolver<>0 entonces
   MESSAGE('Recuerda. El importe a devolver es %1€',nvImporteDevolver)
Si ConfigTpv.Datafono=PAYLINK O ConfigTpv.Datafono=CLEARONE entonces
   Multiforma se filtra TipoDocumento sea TablaTicket.TipoDocumento
   Multiforma se filtra NoDocumento sea TablaTicket.NoDocumento
   Multiforma se filtra TipoPago sea Tarjeta
   Multiforma se filtra Datafono_Resultado <> Autorizada
   Si encuentra registros Multiforma entonces
      Si No Confirma('¿ Realmente deseas continuar con el registro de la factura ?')
         ERROR('Proceso cancelado')
LinVenta se inicializa
LinVenta se filtra TipoDocumento sea Factura
LinVenta se filtra NoDocumento sea TablaTicket.NoDocumento
LinVenta se posiciona primer registro
Repetir
    LinVenta.Impreso=Verdadero
    LinVenta.PrecioVentaImpreso=LinVenta.PrecioVenta
    LinVenta.ImporteLineaImpreso=LinVenta.ImporteLinea
    LinVenta.FechaImpresion=TablaTicket.FechaDocumento
    LinVenta.ImporteIVA=LinVenta.ImporteIVA
    LinVenta.CantidadImpreso=LinVenta.Cantidad
Hasta registro LinVenta sea vacío
RegistrarFactura

HistVenta se inicializa
HistVenta se filtra NoAsignado sea TablaTicket.NoDocumento
Si encuentra registros HistVenta entonces
   HistVenta.ErrorEnvio=Verdadero
   Modificar registro HistVenta
Sino
   ERROR('ERROR Grave, no existe histórico de ticket')
ConfigVenta se posiciona primer registro
Multiforma se inicializa
Multiforma se filtra TipoDocumento sea TablaTicket.TipoDocumento
Multiforma se filtra NoDocumento sea TablaTicket.NoDocumento
Multiforma se filtra TipoPago sea Vale
Si encuentra registros Multiforma entonces
   Repetir
      Si FormaPago.Coge(Multiforma.CodigoForma) Y
         FormaPago.Cumplimentacion=Eleccion de Lista entonces
         TablaBono.Coge(Multiforma.NoBono/Reserva/Abono)
         TablaBono.ImporteLiquidado=TablaBono.ImporteLiquidado+
          Multiforma.Importe
         TablaBono.Pend=TablaBono.ImporteLiquidado<>TablaBono.ImporteInicial
         TablaBono.FechaUltMod=Hoy
         Si TablaBono.Pendiente entonces
            TablaBono.FechaCaducidad=CALCDATE(
             ConfigVentas.FormulaCaducidad,TablaBono.FechaUltMod)
```

```
        Modificar registro TablaBono
        Si TablaBono.Pendiente entonces
            AuxBono se inicializa
            AuxBono se filtra NoDocReg sea TablaBono.NoDocReg
            Si encuentra registros AuxBono entonces
               Caso ConfigTpv.Impresora
                  Negra: InformeBonoN.Filtro(AuxBono)
                         InformeBonoN.UsarOpciones(Falso)
                         Ejecutar InformeBonoN
                 Blanca: InformeBonoB.Filtro(AuxBono)
                         InformeBonoB.UsarOpciones(Falso)
                         Ejecutar InformeBonoB
  Hasta registro Multiforma sea vacío
Multiforma se inicializa
Multiforma se filtra TipoDocumento sea TablaTicket.TipoDocumento
Multiforma se filtra NoDocumento sea TablaTicket.NoDocumento
Multiforma se filtra TipoPago sea Tarjeta
Multiforma se filtra Datafono_Resultado sea Autorizada
Si encuentra registros Multiforma entonces
   Caso ConfigTpv.Impresora
                  Negra: InformeReciboN.Filtro(Multiforma)
                         InformeReciboN.UsarOpciones(Falso)
                         Ejecutar ReciboBonoN
                 Blanca: InformeReciboB.Filtro(Multiforma)
                         InformeReciboB.UsarOpciones(Falso)
                         Ejecutar InformeReciboB
ImprimirRegistro()
VerPromociones(Emision,TablaTicket.NoDocumento)
Si ConfigTpv.ModoTrabajo=On-Line entonces
   Si ServicioWeb.ST_Abrir_Login(ConfigTpv.CodTienda,Config.Password,
                                 Verdadero) entonces
      Si ServicioWeb.TPV_Factura:_Generar(ConfigTpv.CodTienda,Config.Password,
         TablaTicket.NoDocumento,' ',CabVenta.Cliente,HistVenta.Nombre,
         HistVenta.Direccion,HistVenta.CodigoPostal,HistVenta.Ciudad,
         HistVenta.Provincia, HistVenta.Contacto,ConfigTpv.CodTienda,
         HistVenta.FormaPago,Vendedor) entonces
           ConfigTpv.NoFallos=0
           Modificar registro ConfigTpv
           HistVenta se inicializa
           HistVenta se filtra NoAsignado sea TablaTicket.NoDocumento
           Si encuentra registros HistVenta entonces
              HistVenta.ErrorEnvio=Falso
              Modificar registro HistVenta
      Sino
         ConfigTpv.NoFallos=Config.NoFallos+1
         Modificar registro ConfigTpv
         Si ConfigTpv.NoFallos>3 entonces
            ConfigTpv.FechaOffLine=Hoy
            ConfigTpv.HoraOffLine=Hoy
            ConfigTpv.ModoTrabajo=Off-Line
            Modificar registro ConfigTpv
Sino
   ConfigTpv.NoFallos=Config.NoFallos+1
   Modificar registro ConfigTpv
```

```
   Si ConfigTpv.NoFallos>3 entonces
      ConfigTpv.FechaOffLine=Hoy
      ConfigTpv.HoraOffLine=Hoy
      ConfigTpv.ModoTrabajo=Off-Line
      Modificar registro ConfigTpv
```

Código Original

```
IF Vendedor = '' THEN
  ERROR(Text00001);
Cventa.RESET;
Cventa.SETRANGE("Document Type", Cventa."Document Type"::Invoice);
Cventa.SETRANGE("No.", "Nº documento");
Cventa.FIND('-');
IF Cventa."Bal. Account No." <> '' THEN
 ERROR(Text00002+Text00003);

Cventa.VALIDATE(Cventa."Salesperson Code",Vendedor);
Cventa.MODIFY;

"Fecha documento":=WORKDATE;
"Hora documento":=TIME;
MODIFY;

FechaTic := "Fecha documento";

Config.RESET;
Config.SETFILTER(Config."Cód. TPV", "Cód. TPV");
Config.FIND('-');
Config.TESTFIELD("Forma pago EFECTIVO");
Config.TESTFIELD("Forma pago VALES recibidos");

IF ("Cód. tienda" = '') AND ("Cód. TPV" = '') THEN
 ERROR(Text00006);

IF ("Total a pagar" <> "Total venta provisional") AND (Cventa."Nº Entrega/Ticket" = '') THEN
 ERROR(Text00006);

MultiPago.RESET;
MultiPago.SETRANGE("Tipo Documento", "Tipo documento");
MultiPago.SETRANGE("Nº Documento", "Nº documento");
MultiPago.SETRANGE("Tipo pago", MultiPago."Tipo pago"::Vale);

IF MultiPago.FIND('-') THEN
REPEAT
 IF (rcdFormaPago.GET(MultiPago."Cod. Forma Pago")) AND
    (rcdFormaPago."Cumplimentación Nº Vale" = rcdFormaPago."Cumplimentación Nº
Vale"::"Elección de Lista") THEN
 BEGIN
   Bono.GET(MultiPago."Nº bono / Reserva / Abono");
   Bono.TESTFIELD(Pendiente);
 END;
UNTIL MultiPago.NEXT = 0;

PintarTicket := Cventa."Sell-to Customer No." = Config."Cód. cliente TPV";
```

```
nvImporteADevolver:=0;
MultiPago.RESET;
MultiPago.SETRANGE("Tipo Documento", "Tipo documento");
MultiPago.SETRANGE("Nº Documento", "Nº documento");
IF MultiPago.FIND('-') THEN
BEGIN
  SumForPago := 0;
  REPEAT
    IF MultiPago."Tipo pago" = MultiPago."Tipo pago"::Caja THEN
    BEGIN
      SumForPago += MultiPago.Cobrado;
      nvImporteADevolver+=MultiPago.Cambio;
    END
    ELSE
      SumForPago += MultiPago.Importe;
  UNTIL MultiPago.NEXT = 0;
  IF SumForPago <> ("Total a pagar" - "Importe entregado") THEN
    ERROR(Text00007,SumForPago,"Total a pagar" - "Importe entregado");
END ELSE
  ERROR(Text00008);

IF nvImporteADevolver <> 0 THEN
   MESSAGE(Text00009,nvImporteADevolver);

IF ((cduILR.DatafonoHabilitado("Cód. TPV")='PAYLINK') OR
(cduILR.DatafonoHabilitado("Cód. TPV")='CLEARONE')  ) THEN
BEGIN
  MultiPago.RESET;
  MultiPago.SETCURRENTKEY("Tipo Documento","Nº Documento","Tipo pago");
  MultiPago.SETRANGE("Tipo Documento", "Tipo documento");
  MultiPago.SETRANGE("Nº Documento", "Nº documento");
  MultiPago.SETRANGE("Tipo pago",MultiPago."Tipo pago"::Tarjeta);
  MultiPago.SETFILTER(Datafono_Resultado,'<>
%1',MultiPago.Datafono_Resultado::Autorizada);
  IF MultiPago.FIND('-') THEN
     IF NOT CONFIRM(Text00010+Text00011+Text00012,FALSE) THEN
       ERROR(Text00013);
END;

LinVenta.RESET;
LinVenta.SETRANGE("Document Type",LinVenta."Document Type"::Invoice);
LinVenta.SETRANGE(LinVenta."Document No.","Nº documento");
LinVenta.FIND('-');
REPEAT
  LinVenta.Impreso := TRUE;
  LinVenta."Precio venta impreso" := LinVenta."Unit Price";
  LinVenta."Importe línea impreso" := LinVenta."Line Amount";
  LinVenta."Fecha impresión" := "Fecha documento";
  LinVenta."Importe Incl. IVA impreso" := LinVenta."Outstanding Amount";
  LinVenta."Cantidad impreso" := LinVenta.Quantity;
  LinVenta.MODIFY;
UNTIL LinVenta.NEXT = 0;
```

```
COMMIT;
CLEAR(Regis);
Regis.RUN(Cventa);
COMMIT;

CabVentaReg.RESET;
CabVentaReg.SETCURRENTKEY("Pre-Assigned No.");
CabVentaReg.SETRANGE( CabVentaReg."Pre-Assigned No.","Nº documento");
IF CabVentaReg.FIND('-') THEN
BEGIN
  CabVentaReg."Error envío":= TRUE;
  CabVentaReg.MODIFY;
  COMMIT;
 END
 ELSE
  ERROR(Text00014);

ConfVentas.FIND('-');
MultiPago.RESET;
MultiPago.SETRANGE("Tipo Documento", "Tipo documento");
MultiPago.SETRANGE("Nº Documento", "Nº documento");
MultiPago.SETRANGE("Tipo pago", MultiPago."Tipo pago"::Vale);
IF MultiPago.FINDSET() THEN
REPEAT
 IF (rcdFormaPago.GET(MultiPago."Cod. Forma Pago")) AND
   (rcdFormaPago."Cumplimentación Nº Vale" = rcdFormaPago."Cumplimentación Nº
Vale"::"Elección de Lista") THEN
 BEGIN
   Bono.GET(MultiPago."Nº bono / Reserva / Abono");
   Bono.TESTFIELD(Pendiente);
   Bono."Importe liquidado (DL)" := Bono."Importe liquidado (DL)" + MultiPago.Importe;
   Bono.Pendiente := Bono."Importe liquidado (DL)" <> Bono."Importe inicial (DL)";
   Bono."Fecha ult. modificación" := TODAY;
   IF Bono.Pendiente THEN
     Bono."Fecha Caducidad":=CALCDATE(ConfVentas."Fórmula caducidad
bonos",Bono."Fecha ult. modificación");
   Bono.MODIFY;

   IF Bono.Pendiente THEN
   BEGIN
    Auxbono.RESET;
    Auxbono.SETRANGE("Nº documento registrado",Bono."Nº documento registrado");
    IF Auxbono.FINDFIRST THEN;
    IF MultiPago."Cod. Forma Pago" = 'VALEDTO' THEN
    BEGIN
     cduILR.ReimprimePromociones(Auxbono);
    END
    ELSE
    BEGIN
     CASE Config."Impresora Tickets" OF
       Config."Impresora Tickets"::Negra:
         BEGIN
           CLEAR(rptBonoPolyNegro);
           rptBonoPolyNegro.SETTABLEVIEW(Auxbono);
```

```
            rptBonoPolyNegro.USEREQUESTFORM(FALSE);
            rptBonoPolyNegro.RUNMODAL();
           END;
         Config."Impresora Tickets"::Blanca:
           BEGIN
            CLEAR(rptBonoPolyBlanco);
            rptBonoPolyBlanco.SETTABLEVIEW(Auxbono);
            rptBonoPolyBlanco.USEREQUESTFORM(FALSE);
            rptBonoPolyBlanco.RUNMODAL();
           END;
       END;
      END;
   END;
  END;
UNTIL MultiPago.NEXT = 0;
rcdMultiForma.RESET;
rcdMultiForma.SETCURRENTKEY("Tipo Documento","Nº Documento","Nº Linea");
rcdMultiForma.SETRANGE("Tipo Documento","Tipo documento");
rcdMultiForma.SETRANGE("Nº Documento","Nº documento");
rcdMultiForma.SETRANGE("Tipo pago",rcdMultiForma."Tipo pago"::Tarjeta);
rcdMultiForma.SETRANGE(Datafono_Resultado,rcdMultiForma.Datafono_Resultado::Autoriz
ada);
IF rcdMultiForma.FIND('-') THEN
BEGIN
  CASE Config."Impresora Tickets" OF
      Config."Impresora Tickets"::Negra:
        BEGIN
         CLEAR(rptReciboNegro);
         rptReciboNegro.SETTABLEVIEW(rcdMultiForma);
         rptReciboNegro.USEREQUESTFORM(FALSE);
         rptReciboNegro.RUN();
        END;
      Config."Impresora Tickets"::Blanca:
        BEGIN
         CLEAR(rptReciboBlanco);
         rptReciboBlanco.SETTABLEVIEW(rcdMultiForma);
         rptReciboBlanco.USEREQUESTFORM(FALSE);
         rptReciboBlanco.RUN();
        END;
  END;
END;
Rec.PrintRecords(FALSE);
COMMIT;
cduILR.VerPromociones(rcdPromocion."Emision/Consumo"::Emisión,"Nº documento");
IF Config."Modo de Trabajo"= Config."Modo de Trabajo"::"On-Line" THEN
BEGIN
  IF WS.ST_Abrir_Login(Config."Cód. tienda",Config.Password,TRUE) THEN
  BEGIN
    IF WS.TPV_Factura_Generar(Config."Cód. tienda",Config.Password,"Nº documento",",
            Cventa."Sell-to Customer No.",CabVentaReg."Ship-to Name",CabVentaReg."Ship-to
Address" ,
            CabVentaReg."Ship-to Post Code" ,CabVentaReg."Ship-to City"
,CabVentaReg."Ship-to County" ,
```

```
          CabVentaReg."Ship-to Contact",Config."Cód. tienda",CabVentaReg."Payment
Method Code",Vendedor) THEN
   BEGIN
    Config."Nº de fallos":=0;
    Config.MODIFY;

    CabVentaReg.RESET;
    CabVentaReg.SETCURRENTKEY("Pre-Assigned No.");
    CabVentaReg.SETRANGE("Pre-Assigned No.","Nº documento");
    IF CabVentaReg.FIND('-') THEN;

    CabVentaReg."Error envío":= FALSE;
    CabVentaReg.MODIFY;
   END
   ELSE
   BEGIN
     Config."Nº de fallos":=Config."Nº de fallos"+1;
     Config.MODIFY;
     IF Config."Nº de fallos" > 3 THEN
     BEGIN
       Config."Fecha entrada Off-Line":=WORKDATE;
       Config."Hora Entrada Off-Line":=TIME;
       Config."Modo de Trabajo":=Config."Modo de Trabajo"::"Off-Line";
       Config.MODIFY;
     END;
   END;
  END
  ELSE
  BEGIN
    Config."Nº de fallos":=Config."Nº de fallos"+1;
    Config.MODIFY;
    IF Config."Nº de fallos" > 3 THEN
    BEGIN
     Config."Fecha entrada Off-Line":=WORKDATE;
     Config."Hora Entrada Off-Line":=TIME;
     Config."Modo de Trabajo":=Config."Modo de Trabajo"::"Off-Line";
     Config.MODIFY;
    END;
  END;
END;
```

VerPromociones(Tipo,CabVenta.No)

PseudoCódigo

```
TablaPromocion se inicializa
TablaPromocion se filtra Emision/Consumo sea Tipo
TablaPromocion se filtra FechaInicio<=Hoy
TablaPromocion se filtra FechaFin >=Hoy O OD
Si encuentra registros TablaPromocion entonces
    Repetir
       GeneraValeDto(TablaPromocion,CabVenta.No)
    Hasta registro TablaPromocion sea vacío
```

Código Original

```
rcdPromocion.RESET;
rcdPromocion.SETRANGE("Emision/Consumo",PARDirección);
rcdPromocion.SETFILTER("Fecha Inicio",'..%1',WORKDATE);
rcdPromocion.SETFILTER("Fecha Fin",'>=%1| %2',WORKDATE,0D);
IF rcdPromocion.FINDSET THEN
BEGIN
  REPEAT
    GeneraValeDto(rcdPromocion,PARFactura);
  UNTIL rcdPromocion.NEXT=0;
```

GeneraValeDto(TablaPromocion,CabVenta.No)

PseudoCódigo

```
Si TablaPromocion.Emision/Consumo<>Emision entonces
    Salir
nvImporteVale=0
Si No TablaTicket.Coge(TablaTicket.TipoDocumento::Factura,CabVenta.No)
    Salir
nvImporteVale=nvImporteVale+TablaTicket.ImporteEntregado;
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea Numero
ConfigTpv se posiciona primer registro
Multiforma se inicializa
Multiforma se filtra TipoDocumento sea Factura
Multiforma se filtra NoDocumento sea CabVenta.No
Si encuentra registros Multiforma entonces
   Repetir
      Caso Multiforma.TipoPago
        Caja: nvImporteVale=nvImporteVale+Multiforma.Cobrado
        Tarjeta: nvImporteVale=nvImporteVale+Multiforma.Cobrado
        Vale: Si Multiforma.CodFormaPago=ConfigTpv.FormaPagoVales Y
                  TablaBonos.Coge(Multiforma.NoBono/Reserva/Abono Y
                  TablaBonos.Fechacreacion<TablaPromocion.FechaInicio entonces
                    nvImporteVale=nvImporteVale+Multiforma.Cobrado
   Hasta registro Multiforma sea vacío
Si nvImporteVale=0 entonces
    Salir
nvNumVale=TablaPromocion.Prefijo+'-'+CopiaCadena(CabVenta.No,4,3)+'-'+
              CopiaCadena(CabVenta.No,12);
TablaPromoConsumo se inicializa
TablaPromoConsumo se filtra Cod.Promocion sea TablaPromocion.CodPromocion
TablaPromo se posiciona primer registro
TablaBonos se inicializa
TablaBonos.NoDocReg=nvNumVale
TablaBonos.NombreEmpresa=COMPANYNAME
TablaBonos.TipoDocumento=Bono
TablaBonos.ImporteInicial=nvImporteVale
TablaBonos.ImporteBono=nvImporteVale
TablaBonos.Pendiente=Verdadero
TablaBonos.FechaUltiMod=Hoy
TablaBonos.FechaCreacion=Hoy
TablaBonos.FechaCaducidad=TablaPromoConsumo.FechaFin
```

```
TablaBonos.ErrorEnvio=Verdadero
Insertar registro TablaBonos
TablaBonos se inicializa
TablaBonos se filtra NoDocReg sea nvNumVale
Si TablaPromoConsumo.FormulaFechaEmsion<>'' entonces
    TablaBonos.FechaCaducidad=Calcular(TablaPromoConsumo.FormulaFechaEmision
                                       ,Hoy)
Sino
    TablaBonos.FechaCaducidad=TablaPromoConsumo.FechaInicio
```

Código Original

```
IF rcdProm."Emision/Consumo" <> rcdProm."Emision/Consumo"::Emisión THEN
  EXIT;

nvImpVale:=0;
IF NOT rcdTicket.GET(rcdTicket."Tipo documento"::Factura,PARFac) THEN
  EXIT;

Config.RESET;
Config.SETFILTER(Config."Cód. TPV", DameTPV());
Config.FINDFIRST();

nvImpVale+=rcdTicket."Importe entregado";

rcdMulti.RESET;
rcdMulti.SETCURRENTKEY("Tipo Documento","Nº Documento","Nº Linea");
rcdMulti.SETRANGE("Tipo Documento",rcdMulti."Tipo Documento"::Factura);
rcdMulti.SETRANGE("Nº Documento",PARFac);
IF rcdMulti.FINDSET THEN
BEGIN
 REPEAT
  CASE rcdMulti."Tipo pago" OF
      rcdMulti."Tipo pago"::Caja: nvImpVale+=rcdMulti.Cobrado;
      rcdMulti."Tipo pago"::Tarjeta: nvImpVale+=rcdMulti.Cobrado;
      rcdMulti."Tipo pago"::Vale:
       BEGIN
        IF ((rcdMulti."Cod. Forma Pago" = Config."Forma pago VALES emitidos")
AND  (rcdBonos.GET(rcdMulti."Nº bono / Reserva / Abono")) AND
         (rcdBonos."Fecha creación" < rcdProm."Fecha Inicio")) THEN
           nvImpVale+=rcdMulti.Cobrado;
       END;
  END;
 UNTIL rcdMulti.NEXT=0;
END;

IF nvImpVale=0 THEN
    EXIT;

nvNumVale:=rcdProm."Prefijo numeración"+ '-' + COPYSTR(PARFac,4,3) + '-' +
COPYSTR(PARFac,12);
rcdPromConsum.RESET;
rcdPromConsum.SETRANGE("Emision/Consumo",rcdPromConsum."Emision/Consumo"::Con
sumo);
rcdPromConsum.SETRANGE("Cód. Promocion",rcdProm."Cód. Promocion");
```

```
rcdPromConsum.FINDFIRST();

rcdBonos.RESET;
rcdBonos.INIT;
rcdBonos."Nº documento registrado":=nvNumVale;
rcdBonos."Nombre empresa":=COMPANYNAME;
rcdBonos."Tipo documento":=rcdBonos."Tipo documento"::Bono;
rcdBonos."Importe inicial (DL)":=nvImpVale;
rcdBonos."Importe bono (DL)":=nvImpVale;
rcdBonos.Pendiente:=TRUE;
rcdBonos."Fecha ult. modificación":=WORKDATE;
rcdBonos."Fecha creación":=WORKDATE;
rcdBonos."Fecha Caducidad":=rcdPromConsum."Fecha Fin";
rcdBonos."Error Envio":=TRUE;
rcdBonos.INSERT;
COMMIT;

rcdBonos.RESET;
rcdBonos.SETRANGE(rcdBonos."Nº documento registrado",nvNumVale);


CLEAR(rptVale);
rptVale.PonPromocion(rcdProm."Cód. Promocion");

IF rcdPromConsum."Formula Offset Fecha Emisión" <> '' THEN
  rptVale.PonFechaVigencia(CALCDATE(rcdPromConsum."Formula Offset Fecha
Emisión",WORKDATE))
ELSE
  rptVale.PonFechaVigencia(rcdPromConsum."Fecha Inicio");
```

Nº: **TPV157-10/0000001**
Caja: 1
Vendedor:
Fecha: 30/08/10 18:04:34

Ref.	Cant.	Precio	Importe
9-00369	1	24,99	24,99

BARBIE SIRENA CV09

Número de Artículos:1

TOTAL . . . . 24,99€

Desglose de pagos

Tarjeta. . . . . 24,99€

***** I.V.A. incluido *****

No se admiten cambios o reclamaciones sin su ticket.

No se admiten devoluciones de otros centros

Por motivos de higiene, NO SE ADMITIRÁ el cambio ni devolución de artículos textiles (disfraces, zapatos, complementos. Resto de condiciones en el reverso)

GRACIAS POR SU VISITA.

Ilustración 63 Ejemplo de impresión que se genera al facturar el ticket

Ilustración 64 Ejemplo de impresión de vale en caso de que haya promociones activas

Abono Venta

Esta funcionalidad es análoga a la de ticket de venta pero con un par de salvedades: En lo referente a introducir los productos que se quieren abonar se puede realizar de dos maneras: de forma manual, tal y como se hace en el ticket venta, o indicando el ticket que se quiere abonar. En el caso del ticket, el sistema trae todas las líneas pertenecientes a ese ticket, para que luego el dependiente elija que líneas desea abonar. Por otro lado, cuando se va a registrar el abono, se debe indicar una serie de campos obligatorios como son: Motivo de la devolucion, Nombre del cliente y DNI del cliente. Si alguno de esos datos están vacíos, el sistema impide registrar el abono.

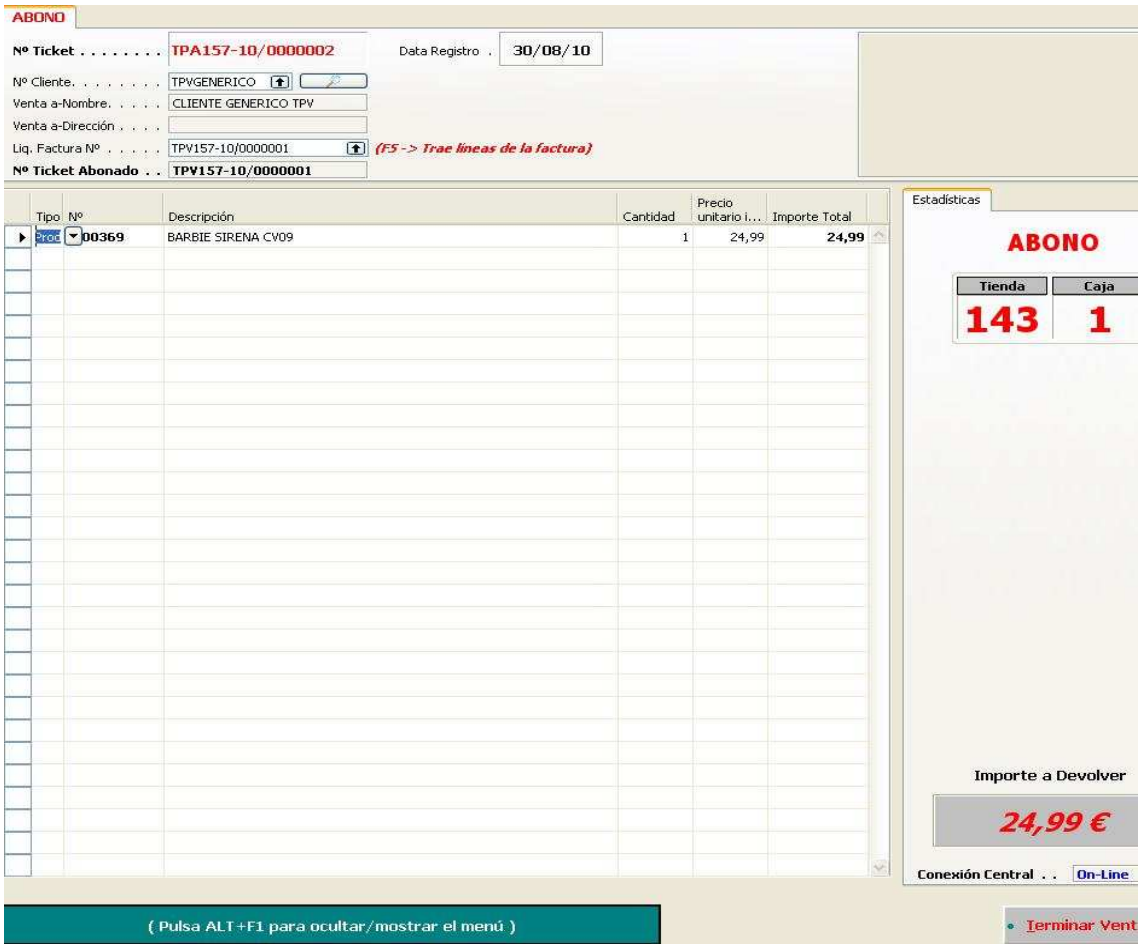


Ilustración 65 Pantalla donde se debe indicar los productos o el ticket que se quiere abonar

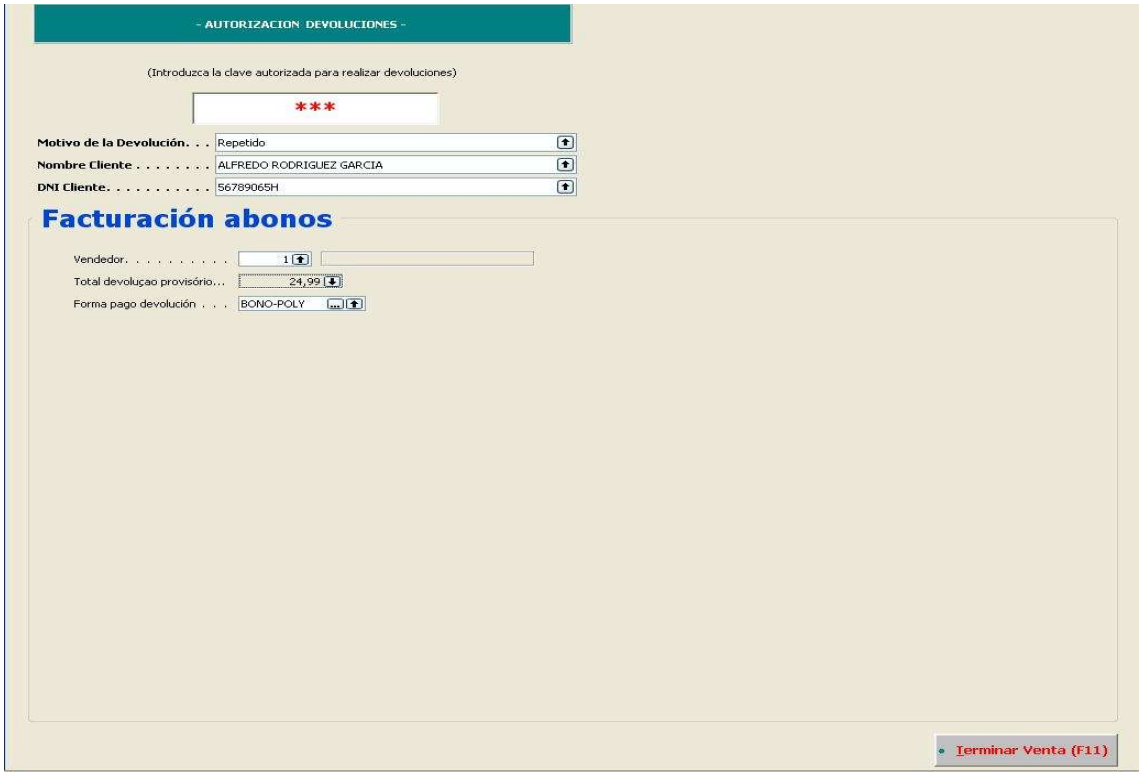


Ilustración 66 Pantalla donde se deben rellenar una serie de datos antes de registrar el abono

Registrar Abono

PseudoCódigo

```
Si Vendedor=’’ entonces
    ERROR(‘Debes indicar el numero de vendedor’)
Si TextoCliente=’’ entonces
    ERROR(‘Debes rellenar el nombre de cliente’)
Si TextoDNI=’’ entonces
    ERROR(‘Debes indicar el DNI del cliente’)
Si CodTienda=’’ Y CodTpv=’’ entonces
    ERROR(‘Abono mal creado’)
Si TextoMotivo=’’ entonces
    ERROR(‘Debes indicar el motivo de la devolucion’)
Sino
    TablaTicket.MotivoDevolucionBono=TextoMotivo
    Modificar registro TablaTicket
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea Numero
ConfigTpv se posiciona en el primer registro
LinVenta se filtra TipoDocumento sea Abono
LinVenta se filtra NoDocumento sea TablaTicket.NoDocumento
LinVenta se filtra Tipo sea Producto
LinVenta se posiciona en el primer registro
Repetir
     Si LinVenta.No=’’ entonces
        ERROR(‘El producto no es valido’)
Hasta registro LinVenta sea vacío
CabVenta se inicializa
CabVenta se filtra TipoDocumento sea Abono
CabVenta.No sea TablaTicket.NoDocumento
CabVenta se posiciona en el primer registro
CabVenta.NombreCliente=TextoCliente
CabVenta.DNICliente=TextoDNI

Si ConfigTpv.Datafono=PAYLINK O ConfigTpv.Datafono=CLEARONE entonces
   Multiforma se filtra TipoDocumento sea TablaTicket.TipoDocumento
   Multiforma se filtra NoDocumento sea TablaTicket.NoDocumento
   Multiforma se filtra TipoPago sea Tarjeta
   Multiforma se filtra Datafono_Resultado <> Autorizada
   Si encuentra registros Multiforma entonces
      Si No Confirma(‘¿ Realmente deseas continuar con el registro de la factura ?’)
         ERROR(‘Proceso cancelado’)
Si TablaTicket.TipoPagoDevol=Tarjeta O TablaTicket.TipoPagoDevol=Caja entonces
    CabVenta.FormaPago= TablaTicket.TipoPagoDevol
Sino
    CabVenta.FormaPago= TablaTicket.TipoPagoDevol
Modificar registro CabVenta
TablaTicket.FechaDocumento=Hoy
TablaTicket.HoraDocumento=Tiempo
Modificar registro TablaTicket
Registrar Abono
Si TablaTicket.FormaPagoDevolucion=ConfigTpv.FormaPagoVales entonces
   TablaBonos se inicializa
   TablaBonos se filtra NoDocReg sea TablaTicket.NoDocumento
```

```
   InformeBono.Filtrar(TablaBono)
   Informe Bono.UsoOpciones(Falso)
   Ejecutar InformeBono
HistoricoAbonos se inicializa
HistoricoAbonos se filtra NoPreasignado sea TablaTicket.NoDocumento
Si no encuentra registros HistoricoAbonos entonces
   ERROR('Error Grave');
HistoricoAbonos.ErrorEnvio=Verdadero
Modificar registro HistoricoAbonos
Multiforma se inicializa
Multiforma se filtra TipoDocumento sea TablaTicket.TipoDocumento
Multiforma se filtra NoDocumento sea TablaTicket.NoDocumento
Multiforma se filtra TipoPago sea Tarjeta
Multiforma se filtra Datafono_Resultado sea Autorizada
Si encuentra registros Multiforma entonces
   Caso ConfigTpv.Impresora
                  Negra: InformeReciboN.Filtro(Multiforma)
                         InformeReciboN.UsarOpciones(Falso)
                         Ejecutar ReciboBonoN
                 Blanca: InformeReciboB.Filtro(Multiforma)
                         InformeReciboB.UsarOpciones(Falso)
                         Ejecutar InformeReciboB
ImprimirRegistro()
Si ConfigTpv.ModoTrabajo=On-Line entonces
   Si ServicioWeb.ST_Abrir_Login(ConfigTpv.CodTienda,Config.Password,
                                 Verdadero) entonces
      Si ServicioWeb.TPV_Abono_Generar(ConfigTpv.CodTienda,Config.Password,
         TablaTicket.NoDocumento,' ',HistoricoAbonos.Cliente,
         HistoricoAbonos Nombre,HistoricoAbonos.Direccion,
         HistoricoAbonos CodigoPostal, HistoricoAbonos.Ciudad,
         HistoricoAbonos.Provincia, HistoricoAbonos.Contacto,ConfigTpv.CodTienda,
         HistVenta.FormaPago,Vendedor,HistoricoAbonos.NoTicketAbonado) entonces
           ConfigTpv.NoFallos=0
           Modificar registro ConfigTpv
           HistVenta se inicializa
           HistVenta se filtra NoAsignado sea TablaTicket.NoDocumento
           Si encuentra registros HistVenta entonces
              HistVenta.ErrorEnvio=Falso
              Modificar registro HistVenta
      Sino
        ConfigTpv.NoFallos=Config.NoFallos+1
        Modificar registro ConfigTpv
        Si ConfigTpv.NoFallos>3 entonces
            ConfigTpv.FechaOffLine=Hoy
            ConfigTpv.HoraOffLine=Hoy
            ConfigTpv.ModoTrabajo=Off-Line
            Modificar registro ConfigTpv
Sino
   ConfigTpv.NoFallos=Config.NoFallos+1
   Modificar registro ConfigTpv
   Si ConfigTpv.NoFallos>3 entonces
      ConfigTpv.FechaOffLine=Hoy
      ConfigTpv.HoraOffLine=Hoy
      ConfigTpv.ModoTrabajo=Off-Line
```

Código Original

```
IF Vendedor = '' THEN
  ERROR(Text00001);

IF txtCliente='' THEN
  ERROR(Text00010);

IF txtDNI='' THEN
  ERROR(Text00011);

IF ("Cód. tienda" = '') AND ("Cód. TPV" = '') THEN
 ERROR(Text00002);

IF txtMotivo='' THEN
  ERROR('Debe especificar el motivo de la devolución')
ELSE BEGIN
 "Motivo Devolución No Bono":=txtMotivo;
 MODIFY;
END;

Config.RESET;
Config.SETFILTER(Config."Cód. TPV",NTpv);
Config.FIND('-');
Config.TESTFIELD("Forma pago VALES emitidos");
LinVenta.SETRANGE("Document Type",LinVenta."Document Type"::"Credit Memo");
LinVenta.SETRANGE(LinVenta."Document No.","Nº documento");
LinVenta.SETRANGE(LinVenta.Type,LinVenta.Type::Item);
LinVenta.FIND('-');
REPEAT
 IF LinVenta."No." = '' THEN
   ERROR(Text00004);
UNTIL LinVenta.NEXT = 0;

Cventa.RESET;
Cventa.SETRANGE("Document Type", Cventa."Document Type"::"Credit Memo");
Cventa.SETRANGE(Cventa."No.","Nº documento");
Cventa.FIND('-');
//-002
Cventa."Nombre Cliente":=txtCliente;
Cventa."DNI CLiente":=txtDNI;
//+002

IF((cduILR.DatafonoHabilitado("Cód.TPV")='PAYLINK')
OR(cduILR.DatafonoHabilitado("Cód. TPV")='CLEARONE')) THEN
BEGIN
 rcdMultiForma.RESET;
 rcdMultiForma.SETRANGE("Tipo Documento", "Tipo documento");
 rcdMultiForma.SETRANGE("Nº Documento", "Nº documento");
 rcdMultiForma.SETRANGE("Tipo pago",rcdMultiForma."Tipo pago"::Tarjeta);
 rcdMultiForma.SETFILTER(Datafono_Resultado,'<>
%1',rcdMultiForma.Datafono_Resultado::Autorizada);
 IF rcdMultiForma.FIND('-') THEN
   IF NOT CONFIRM(Text00005+Text00006+Text00007,FALSE) THEN
     ERROR(Text00008);
```

```
END;

IF ("Tipo pago devolución" = "Tipo pago devolución"::Tarjeta) OR
   ("Tipo pago devolución" = "Tipo pago devolución"::Caja) THEN BEGIN
  Cventa.VALIDATE("Payment Method Code", "Forma pago devolución");
END ELSE BEGIN
  Cventa.VALIDATE("Payment Method Code", "Forma pago devolución");
END;
Cventa.MODIFY;

Rec."Fecha documento":=WORKDATE;
Rec."Hora documento":=TIME;
Rec.MODIFY;

COMMIT;

Regis.RUN(Cventa);
COMMIT;

IF "Forma pago devolución"= Config."Forma pago VALES emitidos" THEN
  BEGIN
    AuxBono.RESET;
    AuxBono.SETRANGE("Nº documento registrado","Nº documento");

    CLEAR(rptBonoPoly);
    rptBonoPoly.SETTABLEVIEW(AuxBono);
    rptBonoPoly.USEREQUESTFORM(FALSE);
    rptBonoPoly.RUNMODAL();
  END;

CabVentareg.RESET;
CabVentareg.SETCURRENTKEY("Pre-Assigned No.");
CabVentareg.SETRANGE( CabVentareg."Pre-Assigned No.","Nº documento");
IF NOT CabVentareg.FIND('-') THEN
   ERROR(Text00009);
CabVentareg."Error envío":= TRUE;
CabVentareg.MODIFY;
COMMIT;

rcdMultiForma.RESET;
rcdMultiForma.SETCURRENTKEY("Tipo Documento","Nº Documento","Nº Linea");
rcdMultiForma.SETRANGE("Tipo Documento","Tipo documento");
rcdMultiForma.SETRANGE("Nº Documento","Nº documento");
rcdMultiForma.SETRANGE("Tipo pago",rcdMultiForma."Tipo pago"::Tarjeta);
rcdMultiForma.SETRANGE(Datafono_Resultado,rcdMultiForma.Datafono_Resultado::Autoriz
ada);
IF rcdMultiForma.FIND('-') THEN
BEGIN
  CASE Config."Impresora Tickets" OF
       Config."Impresora Tickets"::Negra:
         BEGIN
          CLEAR(rptReciboNegro);
          rptReciboNegro.SETTABLEVIEW(rcdMultiForma);
          rptReciboNegro.USEREQUESTFORM(FALSE);
```

```
        rptReciboNegro.RUN();
       END;
     Config."Impresora Tickets"::Blanca:
       BEGIN
        CLEAR(rptReciboBlanco);
        rptReciboBlanco.SETTABLEVIEW(rcdMultiForma);
        rptReciboBlanco.USEREQUESTFORM(FALSE);
        rptReciboBlanco.RUN();
       END;
  END;
END;
Rec.PrintRecords(FALSE);


IF Config."Modo de Trabajo"= Config."Modo de Trabajo"::"On-Line" THEN
BEGIN
   IF WS.ST_Abrir_Login(Config."Cód. tienda",Config.Password,TRUE) THEN
   BEGIN

      IF WS.TPV_Abono_Generar(Config."Cód. tienda",Config.Password,"Nº documento",",
                  CabVentareg."Sell-to Customer No.",CabVentareg."Ship-to
Name",CabVentareg."Ship-to Address" ,
                  CabVentareg."Ship-to Post Code" ,CabVentareg."Ship-to City"
,CabVentareg."Ship-to County" ,
                  CabVentareg."Ship-to Contact",Config."Cód. tienda",CabVentareg."Payment
Method Code",Vendedor,
                  CabVentareg."Nº Ticket Abonado") THEN
       BEGIN
         CabVentareg."Error envío":= FALSE;
         CabVentareg.MODIFY;
       END
       ELSE
       BEGIN
         Config."Nº de fallos":=Config."Nº de fallos"+1;
         Config.MODIFY;
         IF Config."Nº de fallos" > 3 THEN
         BEGIN
           Config."Fecha entrada Off-Line":=WORKDATE;
           Config."Hora Entrada Off-Line":=TIME;
           Config."Modo de Trabajo":=Config."Modo de Trabajo"::"Off-Line";
           Config.MODIFY;
         END;
       END;
   END
   ELSE
   BEGIN
     Config."Nº de fallos":=Config."Nº de fallos"+1;
     Config.MODIFY;
     IF Config."Nº de fallos" > 3 THEN
     BEGIN
       Config."Fecha entrada Off-Line":=WORKDATE;
       Config."Hora Entrada Off-Line":=TIME;
       Config."Modo de Trabajo":=Config."Modo de Trabajo"::"Off-Line";
       Config.MODIFY;
```

```
    END;
  END;
END;
CurrForm.CLOSE;
```

Reservas

Esta funcionalidad consiste en: Si un cliente llega a la tienda a por un producto en concreto y ocurre que en esos momentos la tienda carece de ese producto pero tiene constancia que dentro de poco lo tendrá, el cliente puede reservar dicho producto dejando una señal, así se asegura que cuando llegue a la tienda, uno de ellos está reservado para él.

También se puede dar el caso de que exista mucha demanda de ese producto y el cliente esté muy interesado en él, por lo que puede aumentar la señal que ha dejado con anterioridad para poder tener preferencia sobre el producto.

En lo referente a dejar una señal, el sistema genera un asiento contable pendiente de liquidar y con la forma de pago idéntica a la del ciente, es decir, que puede ser en efectivo, tarjeta o bono. El asiento se liquidará cuando el cliente venga a recoger el producto y se le haga un ticket de venta con los datos de la reserva.

Cuando el cliente va a la tienda a recoger el producto, existe una opción dentro de reservas, que es convertir a ticket. Esta función lo que hace es generar un ticket de venta con los datos de la reserva, es decir, tiene en cuenta la señal que ha dejado para luego cobrarle únicamente lo que queda por pagar.

Existe el caso de que el cliente ya no esté interesado en el producto y quiera cancelar la reserva. Entonces, el vendedor debe ir a la opción de cancelar reserva. En este caso, el sistema liquida el asiento pendiente que se generó cuando se registró la reserva y emite un bono con la cantidad que dejó de señal el cliente y que puede canjear en sucesivas compras.

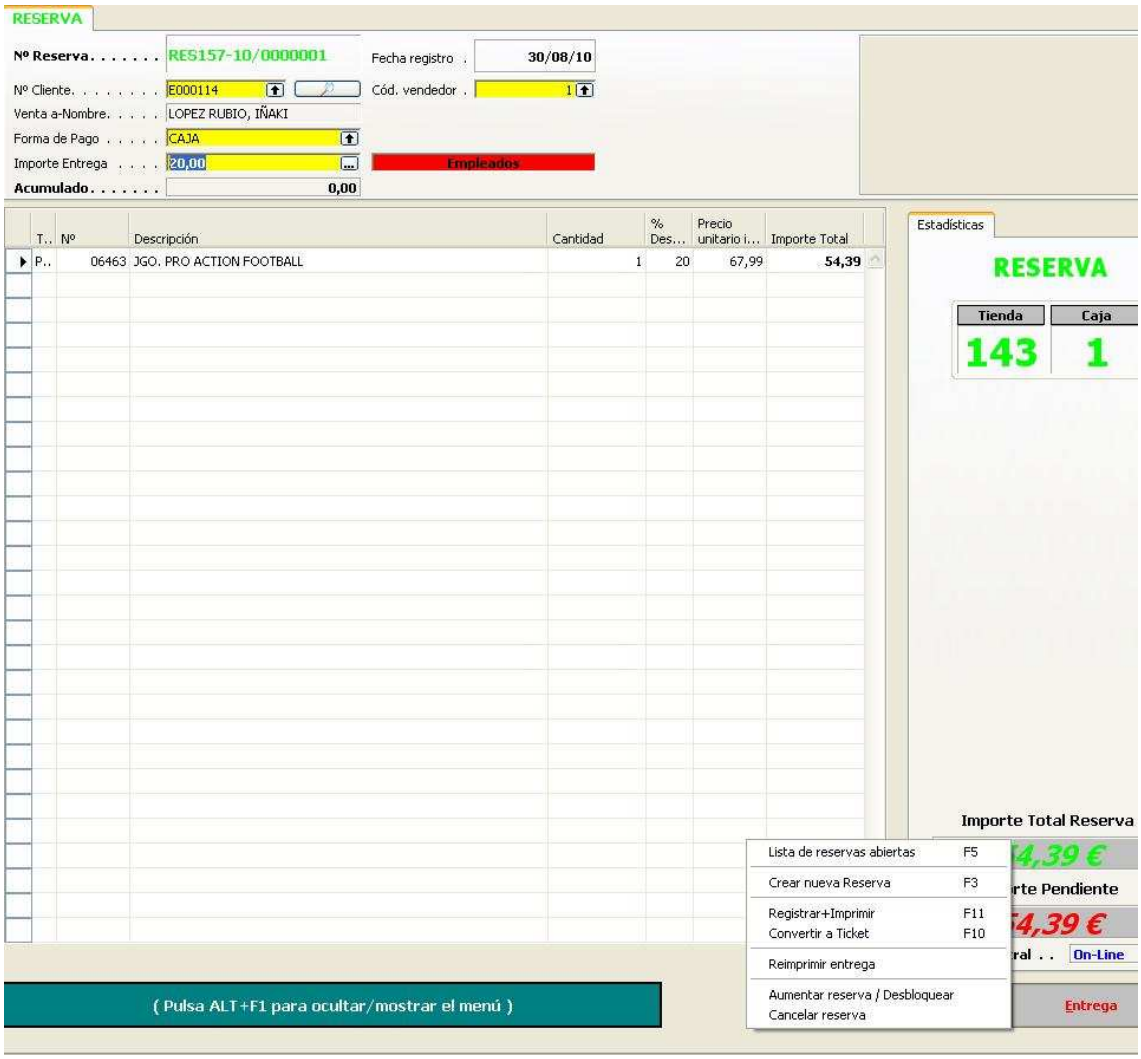


Ilustración 67 Pantalla de las reservas, donde se muestran las funciones que se pueden realizar

CrearReserva

PseudoCódigo

Si No Confirmas(¿Quieres Crear una Nueva Reserva?) entonces
  ERROR('Proceso Cancelado por el usuario')
CabVenta se inicializa
CabVenta.No=''
CabVenta.TipoDocumento=Pedido
CabVenta.HistoricoEntrega=Falso
CabVenta.Entrega=Verdadero
CabVenta.TPV=Verdadero
CabVenta.CodTpv=Numero
CabVenta.CodTienda=CodTienda
CabVenta.Reservarpor=FaltaProducto
Insertar Registro CabVenta
CabVenta se filtra TipoDocumento
CabVenta se filtra No
CabVenta se filtra NoCliente
Message('Reserva %1 creada, CabVenta.No)

Código Original

```
IF NOT CONFIRM(Text00001,FALSE) THEN ERROR(Text00002);
INIT;
"No.":=";
"Document Type":="Document Type"::Order;
"Histórico de entrega":=FALSE;
Entrega := TRUE;
TPV := TRUE;
"Cód. TPV" := numTPV;
"Cód. tienda" := CodTienda;
"Reservas por" := "Reservas por"::"Falta producto";
INSERT(TRUE);

SETRANGE("Document Type");
SETRANGE("No.");
SETRANGE("Sell-to Customer No.");
MESSAGE(Text00003,"No.");
```

Registrar Reserva

PseudoCódigo

```
ComprobarSeñal( )
Si No TablaFormaPago.Coge(CabVenta.CodigoFormaPago) entonces
    ERROR(‘Forma de pago no valida)

Si (ConfigTpv.Datafono=PAYLINK O ConfigTpv.Datafono=CLEARONE) Y
    TablaFormaPago.TipoPago=Tarjeta entonces
    SI No Confirma(¿Estas seguro que deseas registrar el pago con Tarjeta sin utilizar el
        datáfono virtual?) entonces
        ERROR(‘Proceso Cancelado’);
RegistrarSeñal(TRUE);
```

Código Original

```
ComprobarSeñal();
IF NOT rcdformaPago.GET("Payment Method Code") THEN
  ERROR(Text00001);

IF(((cduILR.DatafonoHabilitado("Cód.TPV")='CLEARONE')OR
(cduILR.DatafonoHabilitado("Cód. TPV")='PAYLINK')      )
  AND (rcdformaPago."Tipo Pago"=rcdformaPago."Tipo Pago"::Tarjeta)) THEN
    IF NOT CONFIRM(Text00002+Text00003+Text00004,FALSE) THEN
      ERROR(Text00005);
RegistrarSeñal(TRUE);
```

ComprobarSeñal

PseudoCódigo

```
Si CabVenta.ReservaCancelada entonces
   ERROR(‘Esta reserva ha sido cancelada’);
LinVenta se inicializa
LinVenta se filtra TipoDocumento sea CabVenta.TipoDocumento
LinVenta se filtra NoDocumento sea CabVenta.No
```

```
LinVenta se filtra Cantidad sea distinta de 0
Si no encuentra registros LinVenta entonces
   ERROR(‘Reserva sin productos’)
Si CabVenta.Vendedor=’’ entonces
   ERROR(‘Indicar el Nº Vendedor’)
Si CabVenta.Totalizado entonces
   ERROR(‘Debes primero elegir la opción de Aumentar reserva’)
Si CabVenta.ImporteEntregado=0 entonces
   ERROR(‘Debes indicar el importe que se deja de señal’)
Repetir
   LinVenta.Calcula(‘Total Importe Venta’)
Hasta registro LinVenta sea vacío
Si CabVenta.ImporteEntregado+CabVenta.ImporteAcumuladoReserva=
   LinVenta.TotalImporteVenta entonces
   ERROR(‘No puedes liquidar completamente el importe pendiente de la reserva’)
Si CabVenta.ImporteEntregado+CabVenta.ImporteAcumuladoReserva>
   LinVenta.TotalImporteVenta entonces
   ERROR(‘El importe entregado (%1) no puede superar al importe pendiente (%2).’,
            CabVenta.ImporteEntregado, LinVenta.TotalImporteVenta-
            CabVenta.ImporteAcumuladoReserva)
Si No TablaFormaPago.Coge(CabVenta.CodigoFormaPago) entonces
    ERROR(‘ERROR. Forma de pago no contemplada’)
Si TablaFormaPago.Cumplimentacion=Eleccion de Lista entonces
    TablaBonos.Coge(CabVenta.NoBono)
Si CabVenta.NoContrapartida=’’ entonces
    ERROR(‘HA DE SELECCIONAR UNA FORMA DE PAGO QUE INDIQUE SI
    LA RESERVA ES CONTRA CAJA O CONTRA TARJETAS DE CREDITO).
ConfigTpv se inicializa
ConfigTpv se filtra CodTienda sea CabVenta.CodTienda
ConfigTpv se filtra CodTpv sea CabVenta.CodTpv
ConfigTpv se posiciona en el primer registro
TablaFormaPago se inicializa
TablaFormaPago se filtra Codigo sea CabVenta.CodigoFormaPago
Si no encuentra registros TablaFormaPago entonces
   ERROR(‘No se encuentra la forma de pago’)
TablaCaja se inicializa
TablaCaja se filtra CodTienda sea CabVenta.CodTienda
TablaCaja se filtra CodTpv sea CabVenta.CodTpv
TablaCaja se filtra FechaInicioDia sea CabVenta.FechaRegistro
TablaCaja se filtra Activa sea Verdadero
Si no encuentra registros TablaCaja entonces
   ERROR(‘La Caja está cerrada’)
Si no Confirma(¿Confirma que desea registrar la entrega ?) entonces
   ERROR(‘Proceso Cancelado’)
```

Código Original

```
IF "Reserva cancelada" THEN
  ERROR(Text00001);

LinVenta.RESET;
LinVenta.SETRANGE(LinVenta."Document Type","Document Type");
LinVenta.SETRANGE(LinVenta."Document No.","No.");
LinVenta.SETFILTER(LinVenta.Quantity,'<> 0');
IF NOT LinVenta.FIND('-') THEN
```

```
  ERROR(Text00002);

IF "Salesperson Code" = '' THEN
  ERROR(Text00003);

IF "Reserva cancelada" THEN
 ERROR(Text00004);

IF Totalizado THEN
  ERROR(Text00005);

IF ("Importe Entregado" = 0) THEN
 ERROR(Text00006);

REPEAT
  LinVenta.CALCFIELDS("Total importe venta");
UNTIL (LinVenta.NEXT = 0);

IF ("Importe Entregado" + "Importe acumulado reserva" = LinVenta."Total importe venta")
THEN
    ERROR(Text00007+Text00008);

IF ("Importe Entregado" + "Importe acumulado reserva" > LinVenta."Total importe venta")
THEN
    ERROR(Text00009,"Importe Entregado",LinVenta."Total importe venta"- "Importe
acumulado reserva");

LinVenta.RESET;

IF NOT rcdFormaPago.GET("Payment Method Code") THEN
    ERROR(Text00010);

IF rcdFormaPago."Tipo Pago"=rcdFormaPago."Tipo Pago"::Vale THEN
  TESTFIELD("Nº bono")
ELSE
   TESTFIELD("Nº bono",'');

IF rcdFormaPago."Cumplimentación Nº Vale"=rcdFormaPago."Cumplimentación Nº
Vale"::"Elección de Lista" THEN
BEGIN
   BonosComun.GET("Nº bono");
   BonosComun.TESTFIELD(BonosComun.Pendiente, TRUE);
END;

IF "Bal. Account No." = '' THEN
   ERROR(Text00011);

ConfTPV.RESET;
ConfTPV.SETFILTER("Cód. tienda", "Cód. tienda");
ConfTPV.SETFILTER("Cód. TPV","Cód. TPV");
ConfTPV.FIND('-');

FormPago.RESET;
FormPago.SETRANGE(FormPago.Code,"Payment Method Code");
```

```
IF NOT FormPago.FIND('-') THEN
  ERROR(Text00012,"Payment Method Code");

Caja.RESET;
Caja.SETRANGE("Cód. tienda", "Cód. tienda");
Caja.SETRANGE(Caja."Cód. TPV","Cód. TPV");
Caja.SETRANGE(Caja."Fecha inicio día","Posting Date");
Caja.SETRANGE(Caja.Activa,TRUE);
IF NOT Caja.FIND('-') THEN
  ERROR(Text00013);

IF NOT CONFIRM(Text00014+Text00015,FALSE,WORKDATE,"Importe Entregado")
THEN
 ERROR(Text00016);
```

RegistrarSeñal(Crear)

PseudoCódigo

```
LinDiario se inicializa
LinDiario se filtra NombreLibro sea COBROS
LinDiario se filtra NombreSeccion sea MULTI+CabVenta.CodTpv
Si encuentra registros LinDiario entonces
    Borrar Todos los Registros
LinDiario se inicializa
LinDiario.NombreLibro=’COBROS’
LinDiario.NombreSeccion=’MULTI’+CabVenta.CodTpv
LinDiario.NoLinea=10000
LinDiario.TipoCuenta=Cliente
LinDiario.NoCuenta=CabVenta.Cliente
LinDiario.FechaRegistro=CabVenta.FechaRegistro
LinDiario.TipoDocumento=Pago
LinDiario.NoDocumento=CabVenta.No
LinDiario.CodigoOrigen=’DIACOBROS’
Si CabVenta.TipoContrapartida=Cuenta entonces
    LinDiario.TipoCuentaContra=Cuenta
Sino
    LinDiario.TipoCuentaContra=Cuenta Banco
LinDiario.NoCuentaContra=CabVenta.NoCuentaContra
LinDiario.ImporteAbonado=CabVenta.ImporteEntregado
LinDiario.CodigoFormaPago=CabVenta.CodigoFormaPago
LinDiario.Importe=-(CabVenta.ImporteEntregado)
LinDiario.ImporteContra(Base)=(CabVenta.ImporteEntregado)
LinDiario.ImporteIVA=-(CabVenta.ImporteEntregado)
LinDiario.ImporteContraIVA=-(CabVenta.ImporteEntregado)
LinDiario.Descripcion=’Reserva’+CabVenta.No
Insertar registro LinDiario
LinDiario se inicializa
LinDiario se filtra NombreLibro sea COBROS
LinDiario se filtra NombreSeccion sea MULTI+CabVenta.CodTpv
Si encuentra registros LinDiario entonces
   Registrar Diario
Caso FormaPago.TipoPago
       Tarjeta: CabVenta.TipoPago=Tarjeta
```

```
        Caja: CabVenta.TipoPago=Caja
        Tarjeta: CabVenta.TipoPago=Bono
    Sino
       ERROR('Ha de seleccionar una forma de pago de caja, tarjeta o bono')

CabVenta.Totalizado=Verdadero
CabVenta.ImporteAcumuladoReserva+=CabVenta.ImporteEntregado
Modificar registro CabVenta
TablaTicket se inicializa
TablaTicket se filtra TipoDocumento sea Reserva
TablaTicket se filtra NoDocumento sea CabVenta.No
Si encuentra registros TablaTicket entonces
     TablaTicket.FechaDocumento=CabVenta.FechaDocumento
     TablaTicket.ImporteEntregado+=CabVenta.ImporteEntregado
     Modificar registro TablaTicket
Multiforma se inicializa
Multiforma se filtra TipoDocumento sea Reserva
Multiforma se filtra NoDocumento sea CabVenta.No
Si encuentra ultimo registro Multiforma entonces
    Linea=Multiforma.NoLinea+10000
Sino
    Linea=10000
Si Crear entonces
    Multiforma se inicializa
    Multiforma.TipoDocumento=Reserva
    Multiforma.NoDocumento=CabVenta.No
    Multiforma.NoLinea=Linea
   Multiforma.Importe=CabVenta.ImporteEntregado
   Multiforma.Cobrado=CabVenta.ImporteEntregado
   Multiforma.CodTpv=CabVenta.CodTpv
   Multiforma.CodTienda=CabVenta.CodTienda
   Insertar registro Multiforme
Multiforma.CodFormaPago=CabVenta.CodFormaPago
Multiforma.Registrado=Verdadero
Multiforma.NoDocReg=CabVenta.No
Multiforma.FechaRegistro=CabVenta.FechaRegistro
Multiforma.NoCliente=CabVenta.Cliente
Multiforma.TipoContra=CabVenta.TipoContra
Multiforma.CuentaContra=CabVenta.CuentaContra
Multiforma.TipoPago=FormaPago.TipoPago
Multiforma.NoBono/Reserva/Abono=CabVenta.NoBono
Modificar registro Multiforme
Si CabVenta.CodFormaPago=ConfigTpv.FormaVales entonces
    TablaBono.Coge(CabVenta.NoBono)
    TablaBono.ImporteLiquidado+=CabVenta.ImporteEntregado
    TablaBono.Pendiente=TablaBono.ImporteLiquidado<>TablaBonos.Importe
    TablaBono.NombreEmpresa=COMPANYNAME
    TablaBono.CodCliente=CabVenta.Cliente
    TablaBono.FechaUltMod=CabVenta.FechaRegistro
    Si TablaBonos.Pendiente entonces
        TablaBonos.FechaCaducidad=Calcular(Configventas.Formula,FechaUltMod)
    Modificar registro TablaBonos
   Si TablaBono.Pendiente entonces
            AuxBono se inicializa
```

```
            AuxBono se filtra NoDocReg sea TablaBono.NoDocReg
            Si encuentra registros AuxBono entonces
               Caso ConfigTpv.Impresora
                  Negra: InformeBonoN.Filtro(AuxBono)
                         InformeBonoN.UsarOpciones(Falso)
                         Ejecutar InformeBonoN
                 Blanca: InformeBonoB.Filtro(AuxBono)
                         InformeBonoB.UsarOpciones(Falso)
                         Ejecutar InformeBonoB
Multiforma se inicializa
Multiforma se filtra TipoDocumento sea TablaTicket.TipoDocumento
Multiforma se filtra NoDocumento sea TablaTicket.NoDocumento
Multiforma se filtra TipoPago sea Tarjeta
Multiforma se filtra Datafono_Resultado sea Autorizada
Si encuentra registros Multiforma entonces
   Caso ConfigTpv.Impresora
                  Negra: InformeReciboN.Filtro(Multiforma)
                         InformeReciboN.UsarOpciones(Falso)
                         Ejecutar ReciboBonoN
                 Blanca: InformeReciboB.Filtro(Multiforma)
                         InformeReciboB.UsarOpciones(Falso)
                         Ejecutar InformeReciboB
CabVenta2 se inicializa
CabVenta2 se filtra TipoDocumento Pedido
CabVenta2 se inicializa No sea CabVenta.No
Si no encuentra registros CabVenta2 entonces
    ERROR('No encuentra reserva')
CabVenta2.ErrorEnvio=Verdadero
Modificar registro CabVenta2
Imprimir Recibo
Si ConfigTpv.ModoTrabajo=On-Line entonces
   Si ServicioWeb.ST_Abrir_Login(ConfigTpv.CodTienda,Config.Password,
                                 Verdadero) entonces
      Si ServicioWeb.TPV_Reserva:_Generar(ConfigTpv.CodTienda,Config.Password,
         CabVenta.No,' ',CabVenta.Cliente, CabVenta.Nombre,
         CabVenta.Direccion, CabVenta.CodigoPostal, CabVenta.Ciudad,
         CabVenta.Provincia, CabVenta.Contacto,ConfigTpv.CodTienda,
         CabVenta.FormaPago,. CabVenta Vendedor,CabVenta.NoBono,
          CabVenta.ImporteEntregado)entonces
           ConfigTpv.NoFallos=0
           Modificar registro ConfigTpv
           HistVenta se inicializa
           HistVenta se filtra NoAsignado sea TablaTicket.NoDocumento
           Si encuentra registros HistVenta entonces
              HistVenta.ErrorEnvio=Falso
              Modificar registro HistVenta
      Sino
        ConfigTpv.NoFallos=Config.NoFallos+1
        Modificar registro ConfigTpv
        Si ConfigTpv.NoFallos>3 entonces
            ConfigTpv.FechaOffLine=Hoy
            ConfigTpv.HoraOffLine=Hoy
            ConfigTpv.ModoTrabajo=Off-Line
            Modificar registro ConfigTpv
```

```
  Sino
    ConfigTpv.NoFallos=Config.NoFallos+1
    Modificar registro ConfigTpv
    Si ConfigTpv.NoFallos>3 entonces
      ConfigTpv.FechaOffLine=Hoy
      ConfigTpv.HoraOffLine=Hoy
      ConfigTpv.ModoTrabajo=Off-Line
      Modificar registro ConfigTpv
```

Código Original

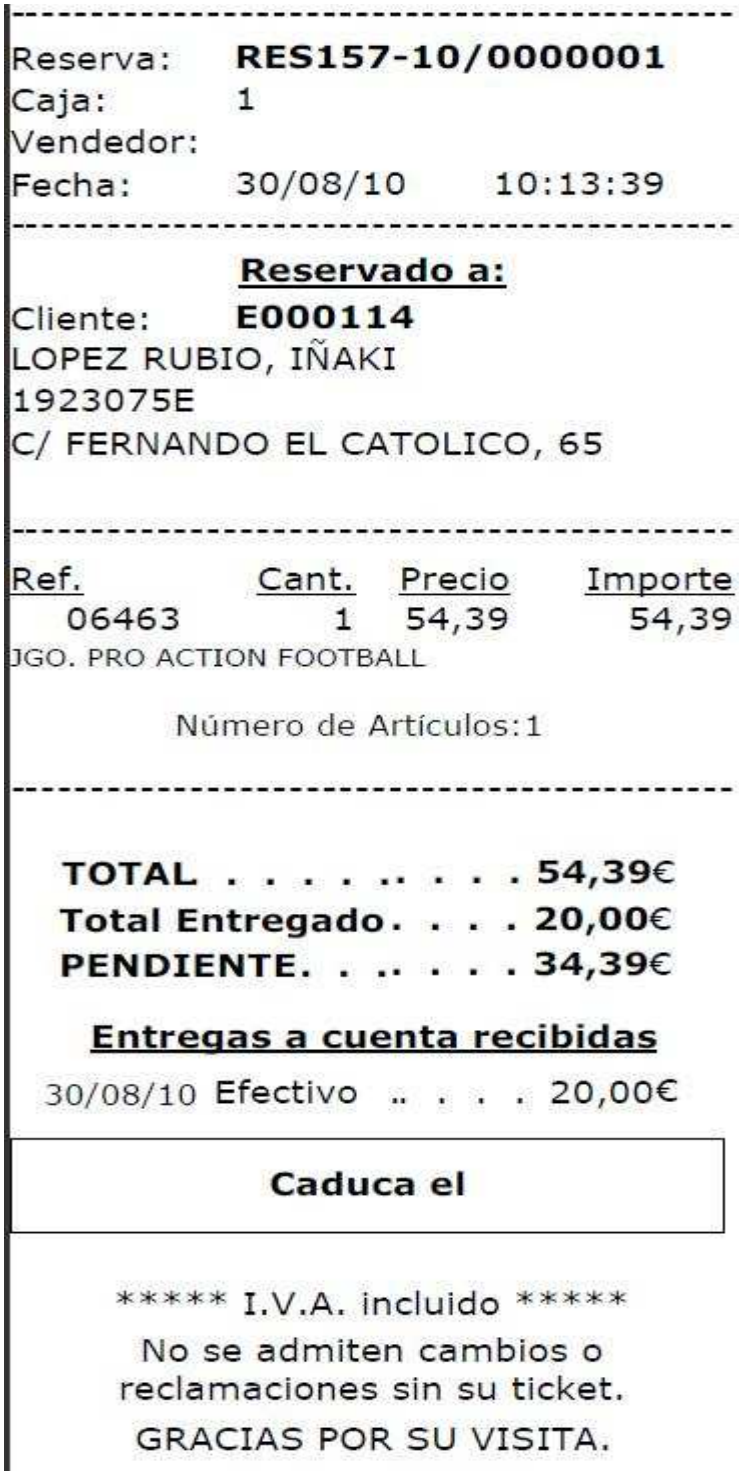
```
LinDiaGen.RESET;
LinDiaGen.SETRANGE("Journal Template Name", 'COBROS');
LinDiaGen.SETRANGE("Journal Batch Name", 'MULTI'+ "Cód. TPV");
IF LinDiaGen.FIND('+') THEN
  LinDiaGen.DELETEALL;
LinDiaGen.INIT;
LinDiaGen."Journal Template Name" := 'COBROS';
LinDiaGen."Journal Batch Name" := 'MULTI'+ "Cód. TPV";
LinDiaGen."Line No." := 10000;
LinDiaGen."Account Type" := LinDiaGen."Account Type"::Customer;
LinDiaGen.VALIDATE("Account Type");
LinDiaGen.VALIDATE("Account No.", "Sell-to Customer No.");
LinDiaGen."Posting Date" := "Posting Date";
LinDiaGen."Document Type" := LinDiaGen."Document Type"::Payment;
LinDiaGen.VALIDATE("Document Type");
LinDiaGen."Document No." := "No.";
LinDiaGen."Source Code" := 'DIACOBROS';
IF "Bal. Account Type" = "Bal. Account Type"::"G/L Account" THEN
  LinDiaGen."Bal. Account Type" := LinDiaGen."Bal. Account Type"::"G/L Account"
ELSE
  LinDiaGen."Bal. Account Type" := LinDiaGen."Bal. Account Type"::"Bank Account";
LinDiaGen.VALIDATE("Bal. Account Type");
LinDiaGen."Bal. Account No." := "Bal. Account No.";
LinDiaGen."Credit Amount" := "Importe Entregado";
LinDiaGen."Payment Method Code" := "Payment Method Code";
LinDiaGen.Amount := -"Importe Entregado";
LinDiaGen."Amount (LCY)" := -"Importe Entregado";
LinDiaGen."VAT Base Amount" := -"Importe Entregado";
LinDiaGen."Bal. VAT Base Amount" := "Importe Entregado";
LinDiaGen."VAT Base Amount (LCY)" := -"Importe Entregado";
LinDiaGen."Bal. VAT Amount (LCY)" := "Importe Entregado";
LinDiaGen.Description := 'Reserva ' + "No.";
LinDiaGen.INSERT;
LinDiaGen.RESET;
LinDiaGen.SETRANGE("Journal Template Name", 'COBROS');
LinDiaGen.SETRANGE("Journal Batch Name", 'MULTI'+ "Cód. TPV");
IF LinDiaGen.FIND('-') THEN;
  CODEUNIT.RUN(CODEUNIT::"Gen. Jnl.-Post",LinDiaGen);
CASE FormPago."Tipo Pago" OF
    FormPago."Tipo Pago"::Tarjeta: "Tipo pago entrega a cuenta" := "Tipo pago entrega a cuenta"::Tarjeta;
    FormPago."Tipo Pago"::Caja:    "Tipo pago entrega a cuenta" := "Tipo pago entrega a cuenta"::Caja;
```

```
      FormPago."Tipo Pago"::Vale:      "Tipo pago entrega a cuenta" := "Tipo pago entrega a
cuenta"::Bono;
  ELSE
      ERROR(Text00001);
  END;

  Rec.LOCKTABLE(TRUE,TRUE);
  Totalizado := TRUE;
  "Importe acumulado reserva" += "Importe Entregado";
  MODIFY;
  Ticket.RESET;
  Ticket.SETRANGE(Ticket."Tipo documento",Ticket."Tipo documento"::Reserva);
  Ticket.SETRANGE(Ticket."Nº documento","No.");
  IF Ticket.FIND('-') THEN BEGIN
     Ticket."Fecha documento":="Posting Date";
     Ticket."Importe entregado" += "Importe Entregado";
     Ticket.MODIFY;
  END;
  MultiPago.RESET;
  MultiPago.SETRANGE("Tipo Documento", MultiPago."Tipo Documento"::Reserva);
  MultiPago.SETRANGE("Nº Documento", "No.");
  IF MultiPago.FIND('+') THEN
    SgteLin := MultiPago."Nº Linea" + 10000
  ELSE
    SgteLin := 10000;
IF PARbCreaMulti THEN
BEGIN
   MultiPago.RESET;
   MultiPago.INIT;
   MultiPago."Tipo Documento" := MultiPago."Tipo Documento"::Reserva;
   MultiPago."Nº Documento"  := "No.";
   MultiPago."Nº Linea" := SgteLin;
   MultiPago.Importe := "Importe Entregado";
   MultiPago.Cobrado := "Importe Entregado";
   MultiPago."Cód. TPV" := "Cód. TPV";
   MultiPago."Cód. tienda" := "Cód. tienda";
   MultiPago.INSERT;
END;
MultiPago."Cod. Forma Pago" := "Payment Method Code";
MultiPago.Registrado := TRUE;
MultiPago."Nº documento registrado" := "No.";
MultiPago."Fecha registro" := "Posting Date";
MultiPago."Nº cliente pago" := "Sell-to Customer No.";
MultiPago."Tipo Contrapartida" := "Bal. Account Type";
MultiPago."Cuenta Contrapartida" := "Bal. Account No.";
MultiPago."Tipo pago" := FormPago."Tipo Pago";
MultiPago."Nº bono / Reserva / Abono" := "Nº bono";
MultiPago.MODIFY;
IF "Payment Method Code" = ConfTPV."Forma pago VALES emitidos" THEN BEGIN
   TESTFIELD("Nº bono");
   BonosComun.GET("Nº bono");
   BonosComun."Importe liquidado (DL)" := BonosComun."Importe liquidado (DL)" +
                                 "Importe Entregado";
```

```
   BonosComun.Pendiente := BonosComun."Importe liquidado (DL)" <>
BonosComun."Importe liquidado (DL)";
   BonosComun."Nombre empresa" := COMPANYNAME;
   BonosComun."Cod. cliente origen" := "Sell-to Customer No.";
   BonosComun."Fecha ult. modificación" := "Posting Date";
   IF BonosComun.Pendiente THEN
     BonosComun."Fecha Caducidad":=CALCDATE(SalesSetup."Fórmula caducidad
bonos",BonosComun."Fecha ult. modificación");
   BonosComun.MODIFY;
   IF BonosComun.Pendiente THEN BEGIN
      Auxbono.RESET;
      Auxbono.SETRANGE("Nº documento registrado",BonosComun."Nº documento
registrado");
      Config.RESET;
      Config.SETFILTER(Config."Cód. TPV", cduILR.DameTPV());
      Config.FIND('-');
      CASE Config."Impresora Tickets" OF
         Config."Impresora Tickets"::Negra: BEGIN
            CLEAR(rptBonoPolyNegro);
            rptBonoPolyNegro.SETTABLEVIEW(Auxbono);
            rptBonoPolyNegro.USEREQUESTFORM(FALSE);
            rptBonoPolyNegro.RUNMODAL();
         END;
         Config."Impresora Tickets"::Blanca: BEGIN
            CLEAR(rptBonoPolyBlanco);
            rptBonoPolyBlanco.SETTABLEVIEW(Auxbono);
            rptBonoPolyBlanco.USEREQUESTFORM(FALSE);
            rptBonoPolyBlanco.RUNMODAL();
         END;
      END;
   END;
END;
rcdMultiForma.RESET;
rcdMultiForma.SETCURRENTKEY("Tipo Documento","Nº Documento","Nº Linea");
rcdMultiForma.SETRANGE("Tipo Documento",rcdMultiForma."Tipo Documento"::Reserva);
rcdMultiForma.SETRANGE("Nº Documento",Rec."No.");
rcdMultiForma.SETRANGE("Nº Linea",LineaParaImprimirReciboDatafon);
rcdMultiForma.SETRANGE("Tipo pago",rcdMultiForma."Tipo pago"::Tarjeta);
rcdMultiForma.SETRANGE(Datafono_Resultado,rcdMultiForma.Datafono_Resultado::Autoriz
ada);
IF rcdMultiForma.FIND('-') THEN
BEGIN
  CASE Config."Impresora Tickets" OF
     Config."Impresora Tickets"::Negra:
       BEGIN
        CLEAR(rptReciboNegro);
        rptReciboNegro.SETTABLEVIEW(rcdMultiForma);
        rptReciboNegro.USEREQUESTFORM(FALSE);
        rptReciboNegro.RUN();
       END;
     Config."Impresora Tickets"::Blanca:
       BEGIN
        CLEAR(rptReciboBlanco);
        rptReciboBlanco.SETTABLEVIEW(rcdMultiForma);
```

```
        rptReciboBlanco.USEREQUESTFORM(FALSE);
        rptReciboBlanco.RUN();
       END;
  END;
END;
CabVenta.RESET;
CabVenta.SETRANGE(CabVenta."Document Type",CabVenta."Document Type"::Order);
CabVenta.SETRANGE( CabVenta."No.","No.");
IF NOT CabVenta.FIND('-') THEN
   ERROR(Text00002);
CabVenta."Error envío":= TRUE;
CabVenta.MODIFY;
COMMIT;
Ticket.PrintRecords(FALSE);

IF ConfTPV."Modo de Trabajo"= ConfTPV."Modo de Trabajo"::"On-Line" THEN
  BEGIN
    IF WS.ST_Abrir_Login(ConfTPV."Cód. tienda",ConfTPV.Password,TRUE) THEN
    BEGIN
        IF          WS.TPV_Reserva_Generar(ConfTPV."Cód.          tienda",ConfTPV.Password,
CabVenta."No.",",
                    CabVenta."Sell-to                Customer                No.",CabVenta."Ship-to
Name",CabVenta."Ship-to Address" ,
                    CabVenta."Ship-to Post Code" ,CabVenta."Ship-to City" ,CabVenta."Ship-to
County" ,
                    CabVenta."Ship-to    Contact",ConfTPV."Cód.    tienda",CabVenta."Payment
Method Code",
                    CabVenta."Salesperson     Code",CabVenta."Nº     bono",CabVenta."Importe
Entregado") THEN
        BEGIN
          CabVenta."Error envío":= FALSE;
          CabVenta.MODIFY;
        END
        ELSE
        BEGIN
          CrearErrorEnvio;
          ConfTPV."Nº de fallos":=ConfTPV."Nº de fallos"+1;
          ConfTPV.MODIFY;
          IF ConfTPV."Nº de fallos" > 3 THEN
          BEGIN
            ConfTPV."Fecha entrada Off-Line":=WORKDATE;
            ConfTPV."Hora Entrada Off-Line":=TIME;
            ConfTPV."Modo de Trabajo":=ConfTPV."Modo de Trabajo"::"Off-Line";
            ConfTPV.MODIFY;
          END;
        END;
    END
    ELSE
    BEGIN
      CrearErrorEnvio;
      ConfTPV."Nº de fallos":=ConfTPV."Nº de fallos"+1;
      ConfTPV.MODIFY;
      IF ConfTPV."Nº de fallos" > 3 THEN
      BEGIN
```

```
      ConfTPV."Fecha entrada Off-Line":=WORKDATE;
      ConfTPV."Hora Entrada Off-Line":=TIME;
      ConfTPV."Modo de Trabajo":=ConfTPV."Modo de Trabajo"::"Off-Line";
      ConfTPV.MODIFY;
    END;
  END;
END
```



Reserva: **RES157-10/0000001**
Caja: 1
Vendedor:
Fecha: 30/08/10 10:13:39

Reservado a:

Cliente: **E000114**
LOPEZ RUBIO, IÑAKI
1923075E
C/ FERNANDO EL CATOLICO, 65

Ref.	Cant.	Precio	Importe
06463	1	54,39	54,39

JGO. PRO ACTION FOOTBALL

Número de Artículos:1

TOTAL . . . . 54,39€
Total Entregado. . . . 20,00€
PENDIENTE. . .. . . . 34,39€

Entregas a cuenta recibidas

30/08/10 Efectivo .. . . . 20,00€

Caduca el

***** I.V.A. incluido *****
No se admiten cambios o reclamaciones sin su ticket.
GRACIAS POR SU VISITA.

Ilustración 68 Ticket que se genera cuando se registra una reserva

Convertir a Ticket

PseudoCódigo

```
Si CabVentaR.ReservaCancelada entonces
   ERROR(‘La reserva fue cancelada.’)
LinVenta se inicializa
LinVenta se filtra TipoDocumento sea CabVentaR.TipoDocumento
LinVenta se filtra NoDocumento sea CabVentaR.No
LinVenta se filtra Cantidad sea <>0
Si no encuentra registros LinVenta entonces
   ERROR(‘Error. No existe ninguna línea en esta reserva’)
Si No Confirmar(Desea generar un ticket mediante esta entrega a cuenta ?,Falso)
   ERROR(‘Proceso abortado por el usuario.’)
Si No CabVenta.Totalizado entonces
   ERROR(‘La entrega aún no ha sido registrada.’)

CrearCabFactura
CrearLineasFactura
CabVenta.Entrega=Falso
CabVenta.HistoricoEntrega=Verdadero
CabVenta.NoEntrega/Ticket=CabVenta.No
Modificar registro CabVenta
Message(‘Se ha creado el ticket nº:%1.’,CabVenta.No)
CabVentaN se inicializa
CabVentaN se filtra TipoDocumento sea CabVenta.TipoDocumento
CabVentaN se filtra No sea CabVenta.No
FormularioTicket.Filtrar(CabVentaN)
FormularioTicket.CogerRegistro(CabVentaN)
FormularioTicket.Ejecutar
```

Código Original

```
IF "Reserva cancelada" THEN
  ERROR(Text00001);


LinVenta.RESET;
LinVenta.SETRANGE(LinVenta."Document Type","Document Type");
LinVenta.SETRANGE(LinVenta."Document No.","No.");
LinVenta.SETFILTER(LinVenta.Quantity,'<> 0');
IF NOT LinVenta.FIND('-') THEN
   ERROR(Text00002);


IF NOT CONFIRM(Text00003, FALSE) THEN
  ERROR(Text00004);
```

```
IF NOT Totalizado THEN
  ERROR(Text00005);

Ventana.OPEN(Text00006+Text00007+Text00008);
Ventana.UPDATE(1,"No.");
HideValidationDialog := TRUE;

CrearCabFactura;
CrearLineasFactura;
Entrega := FALSE;
"Histórico de entrega" := TRUE;
"Nº Entrega/Ticket" := CabVenta."No.";
MODIFY;

Ventana.CLOSE;
MESSAGE('Se ha creado el ticket nº: %1.',CabVenta."No.");

CLEAR(FormTicket);
CabVentaNew.RESET;
CabVentaNew.SETRANGE("Document Type", CabVenta."Document Type");
CabVentaNew.SETRANGE("No.", CabVenta."No.");
FormTicket.SETTABLEVIEW(CabVentaNew);
FormTicket.SETRECORD(CabVentaNew);
FormTicket.RUNMODAL;
```

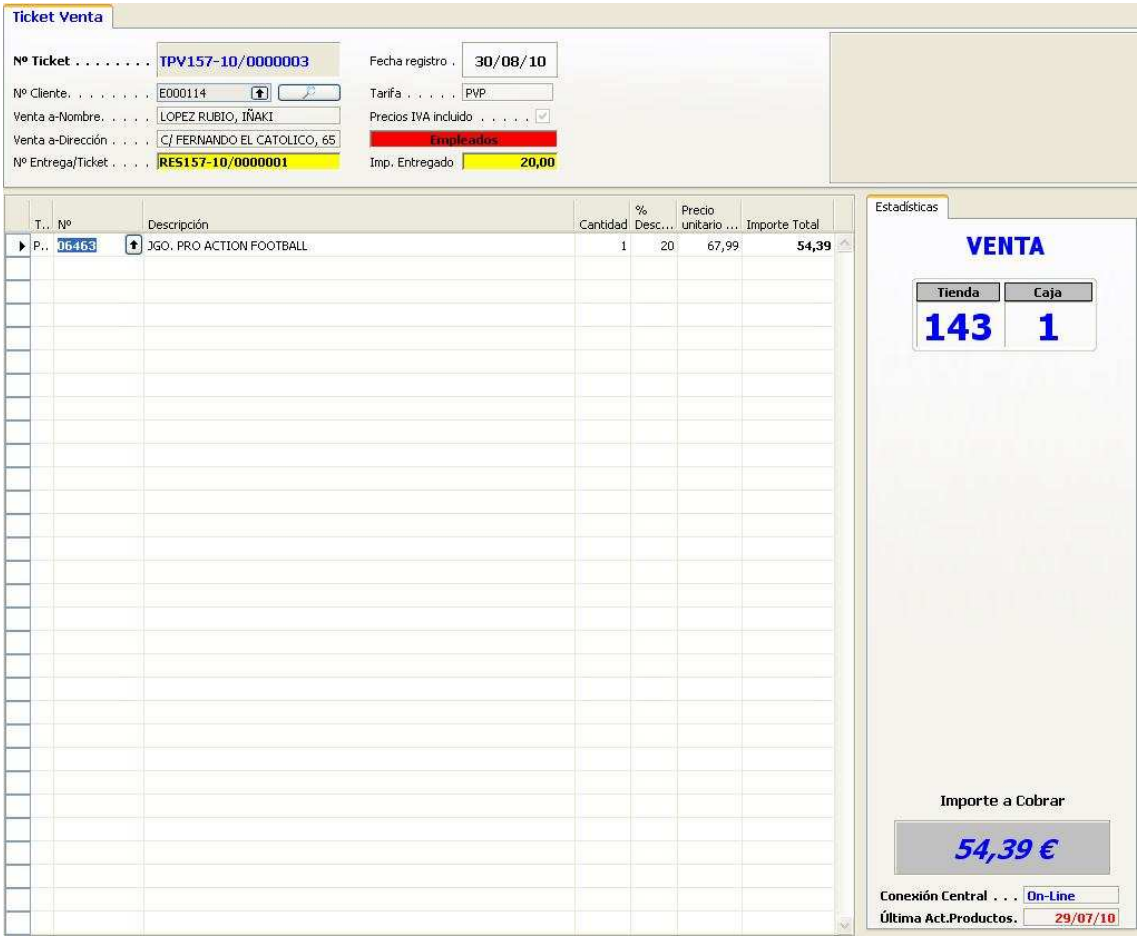


Ilustración 69 Ticket que se genera al ejecutar la opcion Convertir a Ticket de la Reserva

CrearCabFactura

PseudoCódigo

CabVenta.No=’’
CabVenta.ImporteEntregado=CabVentaR.ImporteAcumuladoReserva
CabVenta.FechaRegistro=Hoy
CabVenta.CodTpv=Numero
Insertar registro CabVenta
ConfigTpv se inicializa
ConfigTpv se filtra CodTienda sea CabVentaR.CodTienda
ConfigTpv se filtra CodTpv sea CabVentaR.CodTpv
ConfigTpv se posiciona en el primer registro
CabVenta.CodigoAlmacen=CabVentaR.CodigoAlmacen
CabVenta.Totalizado=Falso
CabVenta.NoEntrega/Ticket=CabVentaR.No
CabVenta.FechaRegistro=Hoy
CabVenta.CodigoFormaPago=ConfigTpv.FormaPagoTpv
CabVenta.ImporteEntregado=CabVentaR.ImporteAcumuladoReserva
Modificar registro CabVenta

Código Original

CabVenta."No." := '';
CabVenta."Importe Entregado" := "Importe acumulado reserva";
CabVenta."Posting Date":=WORKDATE;

```
CabVenta."Cód. TPV":=cduILR.DameTPV();
CabVenta.INSERT(TRUE);

ConfTPV.RESET;
ConfTPV.SETFILTER("Cód. tienda", "Cód. tienda");
ConfTPV.SETFILTER("Cód. TPV","Cód. TPV");
ConfTPV.FIND('-');

CabVenta.VALIDATE("Location Code", "Location Code");
CabVenta.Totalizado := FALSE;
CabVenta."Nº Entrega/Ticket" := "No.";
CabVenta.VALIDATE("Posting Date", WORKDATE);
CabVenta.VALIDATE("Payment Method Code", ConfTPV."Forma pago TPV");
CabVenta."Importe Entregado" := "Importe acumulado reserva";
CabVenta.MODIFY;
```

CrearLineasFactura

PseudoCódigo

```
LinVenta se inicializa
LinVenta se filtra TipoDocumento sea CabVentaR.TipoDocumento
LinVenta se filtra NoDocumento sea CabVentaR.No
LinVenta se filtra Cantidad <>0
Si encuentra registros LinVenta entonces
   Repetir
      LinVentaN se inicializa
      LinVentaN=LinVenta
       LinVentaN.TipoDocumento=CabVenta.TipoDocumento
       LinVentaN.NoDocumento=CabVenta.No
       Insertar registro LinVentaN
       LinVenta.PrecioVentaImpreso=LinVenta.PrecioVenta
       LinVenta.ImporteLineaImpreso=LinVenta.ImporteLinea
       LinVenta.ImporteIncIVAImpreso=LinVenta.ImporteIncIVA
       LinVenta.CantidadImpreso=LinVenta.Cantidad
      Modificar registro LinVenta
      Modificar registro LinVentaN
   Hasta registro LinVenta sea vacío
```

Código Original

```
LinVenta.RESET;
LinVenta.SETRANGE(LinVenta."Document Type","Document Type");
LinVenta.SETRANGE(LinVenta."Document No.","No.");
LinVenta.SETFILTER(LinVenta.Quantity,'<> 0');

IF LinVenta.FIND('-') THEN REPEAT
  Ventana.UPDATE(3,LinVenta."Line No.");

  LinVentaNew.INIT;
  LinVentaNew := LinVenta;
  LinVentaNew."Document Type" := CabVenta."Document Type";
  LinVentaNew."Document No." := CabVenta."No.";
  LinVentaNew.INSERT;
  LinVenta."Precio venta impreso" := LinVenta."Unit Price";
```

```
  LinVenta."Importe línea impreso" := LinVenta."Line Amount";
  LinVenta."Importe Incl. IVA impreso" := LinVenta."Outstanding Amount";
  LinVenta."Cantidad impreso" := LinVenta.Quantity;
  LinVenta.VALIDATE(Quantity,0);
  LinVenta.MODIFY;
  LinVentaNew.MODIFY;
UNTIL LinVenta.NEXT = 0;
```

AumentarSeñal

PseudoCódigo

```
Si CabVenta.HistoricoEntrega entonces
   ERROR('La reserva ya fue liquidada')
Si CabVenta.Totalizado entonces
   CabVenta.CodTpv=Numero
   CabVenta.Totalizado=Falso
   CabVenta.NoBono=''
   CabVenta.ImporteEntregado=0
   Modificar registro CabVenta
   Activar Formulario Reservas
```

Código Original

```
IF "Histórico de entrega" THEN
  ERROR(Text00001+Text00002,"No.");

IF Totalizado THEN BEGIN
  VALIDATE("Posting Date",WORKDATE);
  "Cód. TPV":=cduILR.DameTPV();
  Totalizado := FALSE;
  "Nº bono" := '';
  "Importe Entregado" := 0;
  MODIFY;
  CurrForm.EDITABLE(TRUE);
  CurrForm.SalesLines.EDITABLE(TRUE);
  CurrForm."Payment Method Code".ACTIVATE;
END;
```

Cancelar Reserva

PseudoCódigo

```
Si No Confirma('¿Deseas continuar con la cancelación?') entonces
   ERROR('Proceso Cancelado por el Usuario')
Si CabVenta.HistoricoEntrega entonces
   ERROR('La reserva ya está en el histórico.')
LinVenta se inicializa
LinVenta se filtra TipoDocumento sea CabVenta.TipoDocumento
LinVenta se filtra NoDocumento sea CabVenta.No
LinVenta se filtra Cantidad <>0
Si encuentra registros LinVenta entonces
   Repetir
      LinVenta2=LinVenta
      LinVenta2.CantidadImpreso=LinVenta2.Cantidad
      LinVenta2.Cantidad=0
```

```
        Modificar registro LinVenta2
    Hasta registro LinVenta sea vacío
CabVenta.FechaRegistro=Hoy
CabVenta.HistoricoEntrega=Verdadero
CabVenta.ReservaCancelada=Verdadero
Modificar registro CabVenta
ConfigTpv se inicializa
ConfigTpv se filtra CodTienda sea CabVenta.CodTienda
ConfigTpv se filtra CodTpv sea CabVenta.CodTpv
ConfigTpv se posiciona en el primer registro
Multiforma se inicializa
Multiforma se filtra TipoDocumento sea Reserva
Multiforma se filtra NoDocumento sea CabVenta.No
Si encuentra registros Multiforma entonces
    Linea=Multiforma.NoLinea+10000
Sino
    Linea=10000
Multiforma se inicializa
Multiforma.TipoDocumento=Reserva
Multiforma.NoDocumento=CabVenta.No
Multiforma.NoLinea=Linea
Multiforma.CodFormaPago=ConfigTpv.FormaPagoVales
Multiforma.Importe=CabVenta.ImporteAcumuladoReserva
Multiforma.Cobrado=CabVenta.ImporteAcumuladoReserva
Multiforma.CodTpv=CabVenta.CodTpv
Multiforma.CodTienda=CabVenta.CodTienda
Multiforma.Registrado=Verdadero
Multiforma.NoDocReg=CabVenta.No
Multiforma.FechaRegistro=CabVenta.FechaRegistro
Multiforma.NoClientePago=CabVenta.Cliente
Multiforma.TipoPago=Vale
Multiforma.CancelacionReserva=Verdadero
Insertar registro Multiforma
Si No TablaBonos.Coge(CabVenta.No) entonces
    TablaBonos se inicializa
    TablaBonos.NoDocReg=CabVenta.No
    TablaBonos.Empresa=CompanyName
    TablaBonos.CodCliente=CabVenta.Cliente
    TablaBonos.TipoDocumento=Reserva
    TablaBonos.ImporteReserva=CabVenta.ImporteAcumuladoReserva
    TablaBonos.ImporteInicial=CabVenta.ImporteAcumuladoReserva
    TablaBonos.FechaUltMod=CabVenta.FechaRegistro
    TablaBonos.FechaCreacion=CabVenta.FechaRegistro
    ConfigVentas se posiciona primer registro
    TablaBonos.FechaCaducidad=Calcular(ConfigVenta.FormulaReserva,
                                       TablaBonos.FechaCreacion)
    Insertar registro TablaBonos
TablaBonos.Pendiente=Verdadero
TablaBonos.FechaUltMod=Hoy
TablaBonos.FechaCaducidad=Calcular(ConfigVentas.FormulaBonos,
                                   TablaBonos.FechaCreacion)
Modificar registro TablaBonos
Si TablaBono.Pendiente entonces
        AuxBono se inicializa
```

```
            AuxBono se filtra NoDocReg sea TablaBono.NoDocReg
            Si encuentra registros AuxBono entonces
               Caso ConfigTpv.Impresora
                  Negra: InformeBonoN.Filtro(AuxBono)
                         InformeBonoN.UsarOpciones(Falso)
                         Ejecutar InformeBonoN
                 Blanca: InformeBonoB.Filtro(AuxBono)
                         InformeBonoB.UsarOpciones(Falso)
                         Ejecutar InformeBonoB

Si ConfigTpv.ModoTrabajo=On-Line entonces
 Si ServicioWeb.ST_Abrir_Login(ConfigTpv.CodTienda,Config.Password,
                                    Verdadero) entonces
   CabVenta2 se inicializa
   CabVenta2 se filtra TipoDocumento sea Pedido
   CabVenta2 se filtra No sea CabVenta.No
   Si no encuentra registros CabVenta2 entonces
      ERROR('ERROR Grave, no existe Reserva')
     Si ServicioWeb.TPV_Reserva_Cancelar(ConfigTpv.CodTienda,
        ConfigTpv.Password,CabVenta.No,CabVenta.FechaRegistro,
        ConfigTpv.CodTpv) entonces
       CabVenta2 se inicializa
       CabVenta2 se filtra TipoDocumento sea Pedido
       CabVenta2 se filtra No sea CabVenta.No
       Si encuentra registros CabVenta2 entonces
          CabVenta2.ErrorEnvio=Verdadero
          Modificar registro CabVenta2
          CrearError
       Sino
          ERROR('ERROR Grave, no existe Reserva')
       ConfigTpv.NoFallos=Config.NoFallos+1
       Modificar registro ConfigTpv
       Si ConfigTpv.NoFallos>3 entonces
          ConfigTpv.FechaOffLine=Hoy
          ConfigTpv.HoraOffLine=Hoy
          ConfigTpv.ModoTrabajo=Off-Line
          Modificar registro ConfigTpv
 Sino
   CabVenta2 se inicializa
   CabVenta2 se filtra TipoDocumento sea Pedido
   CabVenta2 se filtra No sea CabVenta.No
   Si encuentra registros CabVenta2 entonces
      CabVenta2.ErrorEnvio=Verdadero
      Modificar registro CabVenta2
      CrearError
   Sino
      ERROR('ERROR Grave, no existe Reserva')
  ConfigTpv.NoFallos=Config.NoFallos+1
   Modificar registro ConfigTpv
   Si ConfigTpv.NoFallos>3 entonces
      ConfigTpv.FechaOffLine=Hoy
      ConfigTpv.HoraOffLine=Hoy
      ConfigTpv.ModoTrabajo=Off-Line
      Modificar registro ConfigTpv
```

```
Sino
   CabVenta2 se inicializa
   CabVenta2 se filtra TipoDocumento sea Pedido
   CabVenta2 se filtra No sea CabVenta.No
   Si encuentra registros CabVenta2 entonces
      CabVenta2.ErrorEnvio=Verdadero
      Modificar registro CabVenta2
      CrearError
   Sino
      ERROR(‘ERROR Grave, no existe Reserva’)
Mensaje(‘Cancelación realizada’)
```

Código Original

```
IF NOT CONFIRM(Text00001+Text00002,FALSE,"Importe acumulado reserva") THEN
  ERROR(Text00003);

IF "Histórico de entrega" THEN
 ERROR(Text00006);

LinVenta.RESET;
LinVenta.SETRANGE(LinVenta."Document Type","Document Type");
LinVenta.SETRANGE(LinVenta."Document No.","No.");
LinVenta.SETFILTER(LinVenta.Quantity,'<> 0');
IF LinVenta.FIND('-') THEN REPEAT
 LinVenta2 := LinVenta;
 LinVenta2."Cantidad impreso" := LinVenta2.Quantity;
 LinVenta2.VALIDATE(Quantity,0);
 LinVenta2.MODIFY;
UNTIL LinVenta.NEXT = 0;

VALIDATE("Posting Date",WORKDATE);
"Histórico de entrega" := TRUE;
"Reserva cancelada" := TRUE;
MODIFY;

ConfTPV.RESET;
ConfTPV.SETFILTER("Cód. tienda", "Cód. tienda");
ConfTPV.SETFILTER("Cód. TPV","Cód. TPV");
ConfTPV.FIND('-');

MultiPago.RESET;
MultiPago.SETRANGE("Tipo Documento", MultiPago."Tipo Documento"::Reserva);
MultiPago.SETRANGE("Nº Documento", "No.");
IF MultiPago.FIND('+') THEN
 SgteLin := MultiPago."Nº Linea" + 10000
ELSE
 SgteLin := 10000;

MultiPago.RESET;
MultiPago.INIT;
MultiPago."Tipo Documento" := MultiPago."Tipo Documento"::Reserva;
MultiPago."Nº Documento"  := "No.";
MultiPago."Nº Linea" := SgteLin;
MultiPago."Cod. Forma Pago" := ConfTPV."Forma pago VALES emitidos";
```

```
MultiPago.Importe := "Importe acumulado reserva";
MultiPago.Cobrado := "Importe acumulado reserva";

MultiPago."Cód. TPV" := "Cód. TPV";
MultiPago."Cód. tienda" := "Cód. tienda";
MultiPago.Registrado := TRUE;
MultiPago."Nº documento registrado" := "No.";
MultiPago."Fecha registro" := "Posting Date";
MultiPago."Nº cliente pago" := "Sell-to Customer No.";
MultiPago."Tipo pago" := MultiPago."Tipo pago"::Vale;
MultiPago."Cancelación reserva" := TRUE;
MultiPago.INSERT;

IF NOT BonosComun.GET("No.") THEN
BEGIN
  BonosComun.RESET;
  BonosComun.INIT;
  BonosComun."Nº documento registrado" := "No.";
  BonosComun."Nombre empresa" := COMPANYNAME;
  BonosComun."Cod. cliente origen" := "Sell-to Customer No.";
  BonosComun."Tipo documento" := BonosComun."Tipo documento"::Reserva;
  BonosComun."Importe reserva (DL)" := "Importe acumulado reserva";
  BonosComun."Importe inicial (DL)" := "Importe acumulado reserva";
  BonosComun."Fecha ult. modificación" := "Posting Date";
  BonosComun."Fecha creación" := "Posting Date";
  SalesSetup.FIND('-');
  BonosComun."Fecha Caducidad" := CALCDATE(SalesSetup."Fórmula caducidad
reservas",BonosComun."Fecha creación");
  BonosComun.INSERT;
END;

BonosComun.Pendiente := TRUE;
BonosComun."Fecha ult. modificación":=WORKDATE;
SalesSetup.FIND('-');
BonosComun."Fecha Caducidad" := CALCDATE(SalesSetup."Fórmula caducidad
bonos",BonosComun."Fecha creación");
BonosComun.MODIFY;
  IF BonosComun.Pendiente THEN
  BEGIN
    Auxbono.RESET;
    Auxbono.SETRANGE("Nº documento registrado",BonosComun."Nº documento
registrado");

    Config.RESET;
    Config.SETFILTER(Config."Cód. TPV", cduILR.DameTPV());
    Config.FIND('-');

    CASE Config."Impresora Tickets" OF
      Config."Impresora Tickets"::Negra:
        BEGIN
          CLEAR(rptBonoPolyNegro);
          rptBonoPolyNegro.SETTABLEVIEW(Auxbono);
          rptBonoPolyNegro.USEREQUESTFORM(FALSE);
```

```
        rptBonoPolyNegro.RUNMODAL();
      END;
    Config."Impresora Tickets"::Blanca:
      BEGIN
        CLEAR(rptBonoPolyBlanco);
        rptBonoPolyBlanco.SETTABLEVIEW(Auxbono);
        rptBonoPolyBlanco.USEREQUESTFORM(FALSE);
        rptBonoPolyBlanco.RUNMODAL();
      END;
  END;
END;
IF ConfTPV."Modo de Trabajo"= ConfTPV."Modo de Trabajo"::"On-Line" THEN BEGIN
  IF WS.ST_Abrir_Login(ConfTPV."Cód. tienda",ConfTPV.Password,TRUE) THEN BEGIN
      CabVenta.RESET;
      CabVenta.SETRANGE(CabVenta."Document                Type",CabVenta."Document
Type"::Order);
      CabVenta.SETRANGE( CabVenta."No.","No.");
      IF NOT CabVenta.FIND('-') THEN
            ERROR(Text00007);

      IF    NOT    WS.TPV_Reserva_Cancelar(ConfTPV."Cód.    tienda",ConfTPV.Password,
CabVenta."No.","Posting Date","Cód. TPV") THEN BEGIN
          CabVenta.RESET;
          CabVenta.SETRANGE(CabVenta."Document                Type",CabVenta."Document
Type"::Order);
          CabVenta.SETRANGE(CabVenta."No.","No.");
          IF CabVenta.FIND('-') THEN BEGIN
              CabVenta."Error envío":= TRUE;
              CabVenta.MODIFY;
              CrearErrorCancelar;
          END ELSE
            ERROR(Text00007);

          ConfTPV."Nº de fallos":=ConfTPV."Nº de fallos"+1;
          ConfTPV.MODIFY;
          IF ConfTPV."Nº de fallos" > 3 THEN BEGIN
              ConfTPV."Fecha entrada Off-Line":=WORKDATE;
              ConfTPV."Hora Entrada Off-Line":=TIME;
              ConfTPV."Modo de Trabajo":=ConfTPV."Modo de Trabajo"::"Off-Line";
              ConfTPV.MODIFY;
          END;
      END;
  END ELSE BEGIN
       CabVenta.RESET;
       CabVenta.SETRANGE(CabVenta."Document                Type",CabVenta."Document
Type"::Order);
       CabVenta.SETRANGE(CabVenta."No.","No.");
       IF CabVenta.FIND('-') THEN BEGIN
          CabVenta."Error envío":= TRUE;
          CabVenta.MODIFY;
          CrearErrorCancelar;
       END ELSE
         ERROR(Text00007);
```

```
        ConfTPV."Nº de fallos":=ConfTPV."Nº de fallos"+1;
        ConfTPV.MODIFY;
        IF ConfTPV."Nº de fallos" > 3 THEN BEGIN
            ConfTPV."Fecha entrada Off-Line":=WORKDATE;
            ConfTPV."Hora Entrada Off-Line":=TIME;
            ConfTPV."Modo de Trabajo":=ConfTPV."Modo de Trabajo"::"Off-Line";
            ConfTPV.MODIFY;
        END;
    END;
END ELSE BEGIN
  CabVenta.RESET;
  CabVenta.SETRANGE(CabVenta."Document Type",CabVenta."Document Type"::Order);
  CabVenta.SETRANGE(CabVenta."No.","No.");
  IF CabVenta.FIND('-') THEN BEGIN
      CabVenta."Error envío":= TRUE;
      CabVenta.MODIFY;
      CrearErrorCancelar;
  END ELSE
    ERROR(Text00007);
END;
MESSAGE(Text00008);
```

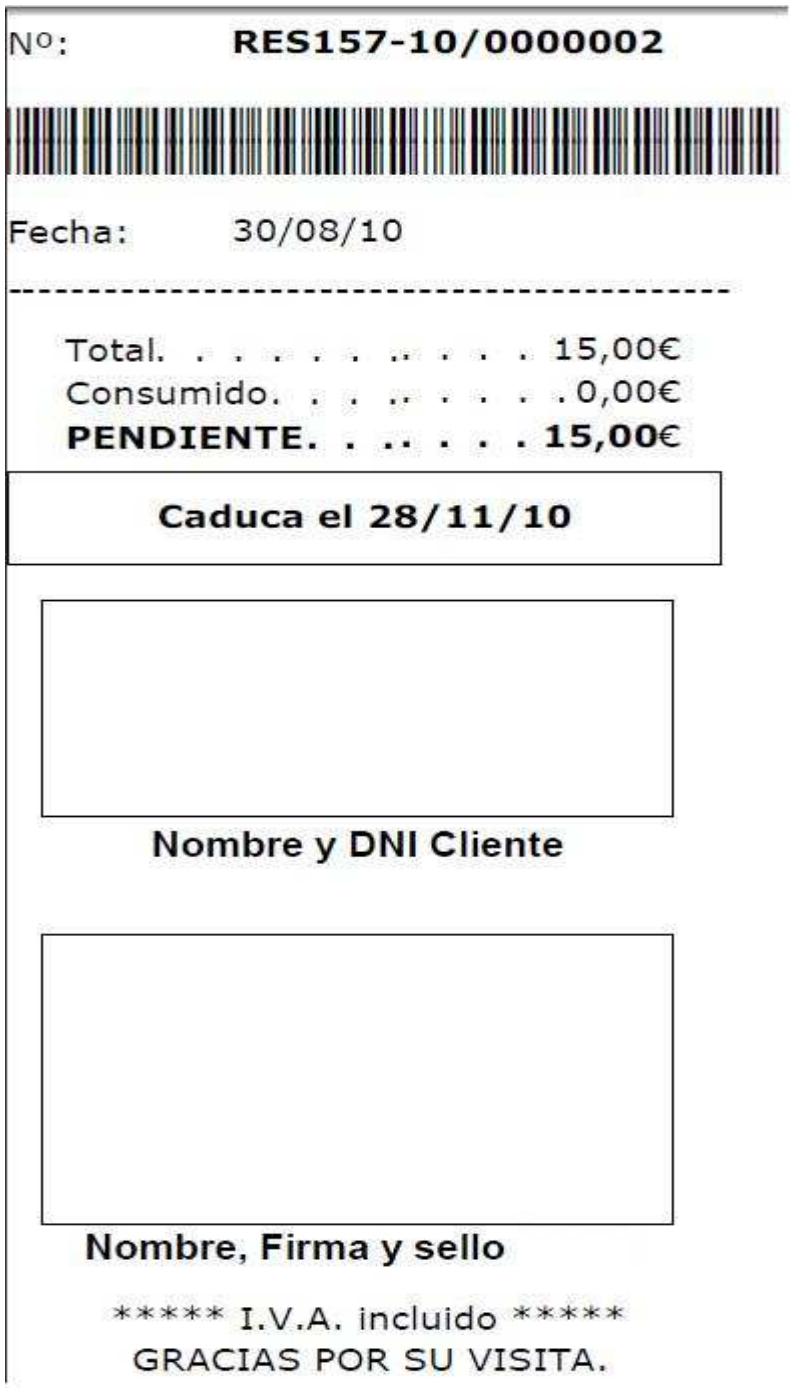


Ilustración 70 Ejemplo de impresión de bono que sale al cancelar la reserva

Traer Bono Central

En esta funcionalidad, cuando el cliente quiere pagar parte de la venta con un bono que se le generó en su momento. Para ello, el vendedor debe clicar en el formulario de Traer Bono Central e indicar que bono quiere traerse.

Una vez tecleado el bono, el sistema evoca a una función mediante servicio web. Esta llamada devuelve error, en caso de no existir, o devuelve la información del bono. Si

recibe información, el sistema incorpora ese bono a la base de datos para que el vendedor pueda utilizarlo sin ningún tipo de problema.

PseudoCódigo

```
Si NumDoc=’’ entonces
    ERROR(‘Ha de introducir un nº de vale.’)
ConfigTpv se inicializa
Si No ConfigTpv se posiciona en el primer registro entonces
   ERROR(‘No existe configuración tpv’)
Si ConfigTpv.ModoTrabajo=On-Line entonces
    Si ServicioWeb.Abrir_Login(ConfigTpv.CodTienda,ConfigTpv.Pass,Verdadero)
       Bono+=ServicioWeb.Traer_Pago_Central(ConfigTpv.CodTienda,
                ConfigTpv.Pass,NumDoc)
       Si Bono<>0 entonces
          Mensaje(‘Se ha creado el vale %1.’,NumDoc)
          Cerrar Formulario
      Sino
          ERROR(‘No se ha podido conectar con la central.’)
Sino
   ERROR(‘No podemos conectar con central, el modo de trabajo es en Off-line.’)
```

Código Original

```
IF NumDoc = '' THEN
  ERROR(Text00001);

  Config.RESET;
  IF NOT Config.FINDFIRST THEN
    ERROR(Text00002);
  IF Config."Modo de Trabajo"= Config."Modo de Trabajo"::"On-Line" THEN BEGIN
     IF WS.ST_Abrir_Login(Config."Cód. tienda",Config.Password,TRUE) THEN
       BEGIN
         NumCab+=WS.TPV_Traer_Pago_Central(Config."Cód.
                    tienda",Config.Password,NumDoc);
         IF NumCab <> 0 THEN
         BEGIN
           MESSAGE(Text00003,NumDoc);
           CurrForm.CLOSE;
         END;
       END ELSE
         ERROR(Text00004);
  END ELSE
    ERROR(Text00005);
```

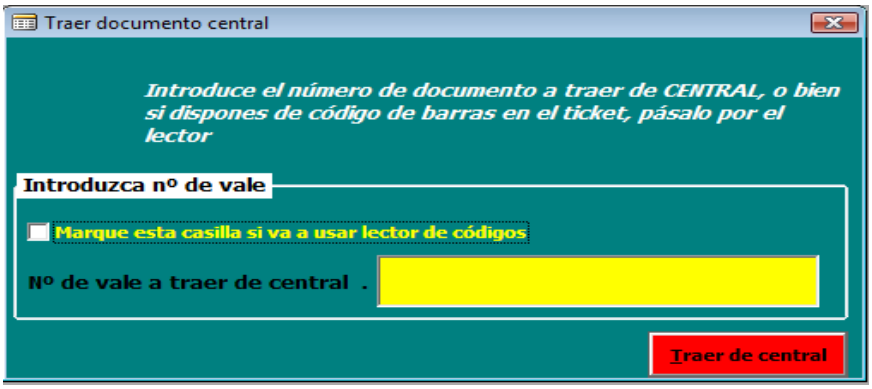


Ilustración 71 Formulario para traer bono de Central

Disponibilidad de Producto por Tienda

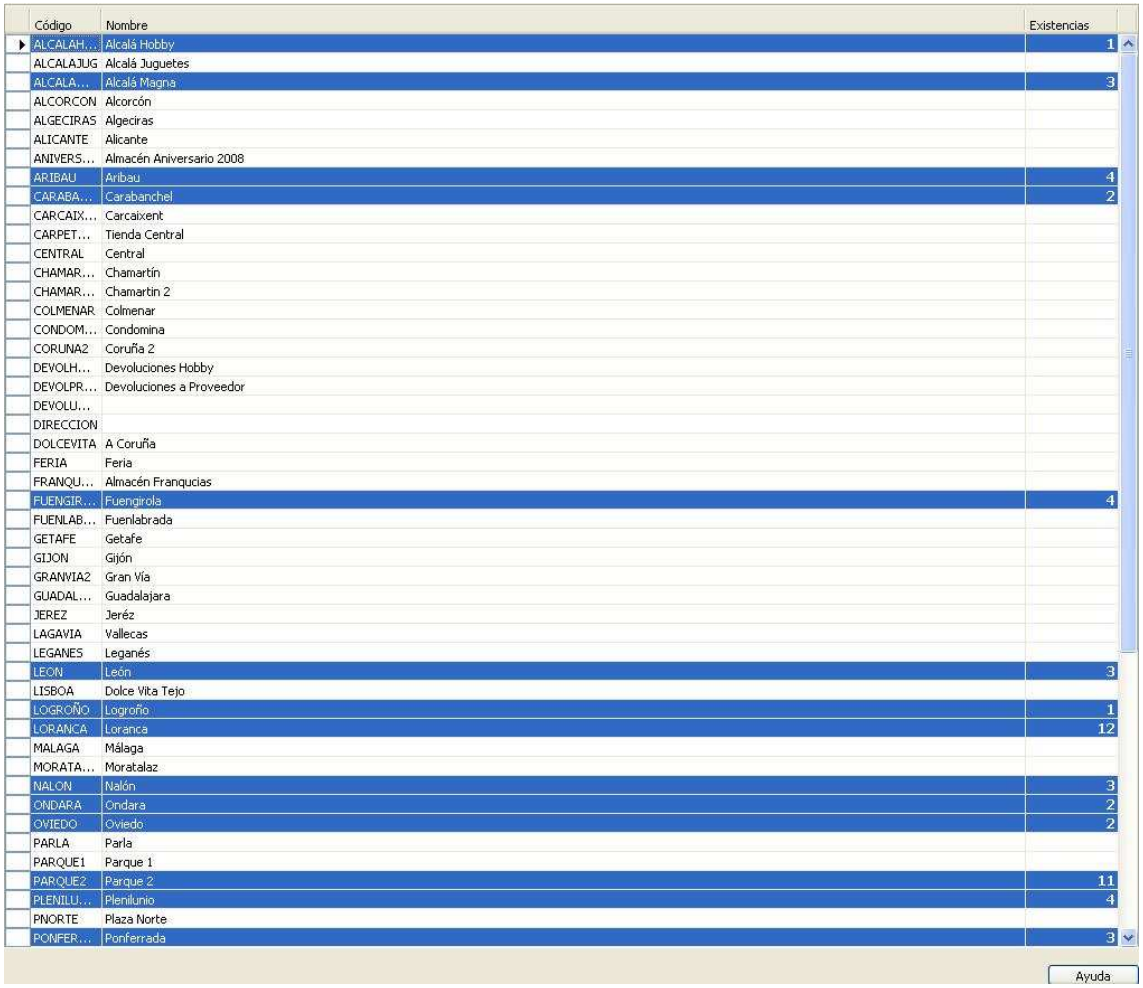
Esta funcionalidad permite saber al personal de tienda qué stock hay en cada tienda de un determinado producto. Esto es útil por si la tienda necesita, de manera urgente, un producto y gracias a esto puede saber qué tienda lo tiene para pedir que se lo envíe y así no perder la venta.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NumTpv
Si no encuentra registros ConfigTpv entonces
   ERROR('No existe configuración tpv')
Si ServicioWeb.TPV_Consulta_Stock(ConfigTpv.CodTienda,ConfigTpv.Password,
    TablaProducto.No,Tienda)
     FormularioTiendas.CogeRegistro(Tienda)
     FormularioTiendas.Ejecutar
Sino
     ERROR('No se ha podido conectar con Central')
```

Código Original

```
Conf.RESET;
  Conf.SETFILTER(Conf."Cód. TPV",NTPV);
  IF NOT Conf.FIND('-') THEN
    ERROR(Text00001);
  IF    WS.TPV_Consulta_stock(Conf."Cód.    tienda",Conf.Password,"No.",Tienda)    THEN
BEGIN
    Formulario.SETRECORD(Tienda);
    Formulario.RUN;
  END
  ELSE
   ERROR(Text00002);
```



Código	Nombre	Existencias
ALCALAH...	Alcalá Hobby	1
ALCALAJUG	Alcalá Juguetes	
ALCALA...	Alcalá Magna	3
ALCORCON	Alcorcón	
ALGECIRAS	Algeciras	
ALICANTE	Alicante	
ANIVERS...	Almacén Aniversario 2008	
ARIBAU	Aribau	4
CARABA...	Carabanchel	2
CARCAIX...	Carcaixent	
CARPET...	Tienda Central	
CENTRAL	Central	
CHAMAR...	Chamartín	
CHAMAR...	Chamartin 2	
COLMENAR	Colmenar	
CONDOM...	Condomina	
CORUNA2	Coruña 2	
DEVOLH...	Devoluciones Hobby	
DEVOLPR...	Devoluciones a Proveedor	
DEVOLU...		
DIRECCION		
DOLCEVITA	A Coruña	
FERIA	Feria	
FRANQU...	Almacén Franquicias	
FUENGIR...	Fuengirola	4
FUENLAB...	Fuenlabrada	
GETAFE	Getafe	
GIJON	Gijón	
GRANVIA2	Gran Vía	
GUADAL...	Guadalajara	
JEREZ	Jeréz	
LAGAVIA	Vallecas	
LEGANES	Leganés	
LEON	León	3
LISBOA	Dolce Vita Tejo	
LOGROÑO	Logroño	1
LORANCA	Loranca	12
MALAGA	Málaga	
MORATA...	Moratalaz	
NALON	Nalón	3
ONDARA	Ondara	2
OVIEDO	Oviedo	2
PARLA	Parla	
PARQUE1	Parque 1	
PARQUE2	Parque 2	11
PLENILU...	Plenilunio	4
PNORTE	Plaza Norte	
PONFER...	Ponferrada	3

Ayuda

Ilustración 72 Formulario que se muestra al consultar la disponibilidad del producto por tienda

Cliente

En este caso, existe un formulario donde se rellenan una serie de datos de un cliente para darle de alta en la base de datos. Una vez rellenados todos los datos, el sistema, de manera automática, inserta ese cliente en la base de datos. A continuación, el personal de tienda debe clicar en la opción enviar, para que ese cliente quede centralizado en la Central y cuando se genere una venta a ese cliente, no de error por no estar centralizado.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NumTpv
Si no encuentra registros ConfigTpv entonces
   ERROR('No existe configuración tpv')
Sino
  Si ConfigTp.ModoTrabajo=On-Line entonces
     Si ServicioWeb.TPV_Alta_Clientes(ConfigTpv.CodTienda,ConfigTpv.Password,
        TablaCliente.No,TablaCliente.Nombre,TablaCliente.Direccion,
        TablaCliente.CodigoPostal,TablaCliente.Ciudad,TablaCliente.Provincia,
```

```
        TablaCliente.Telefono,TablaCliente.Fax,TablaCliente.CodFormaPago,
        TablaCliente.CodigoVendedor,TablaCliente.CodTerminosPago,
        TablaCliente.GrupoRegistroCliente, TablaCliente.GrupoRegistroProducto,
        TablaCliente.GrupoRegistroIVA) entonces

          TablaCliente.ErrorEnvio=Falso
          TablaCliente.EnvioCentral=Verdadero
          Modificar registro TablaCliente
    Sino
         TablaCliente.ErrorEnvio=Verdadero
          TablaCliente.EnvioCentral=Falso
          Modificar registro TablaCliente
          ConfigTpv.NoFallos=Config.NoFallos+1
          Modificar registro ConfigTpv
          Si ConfigTpv.NoFallos>3 entonces
            ConfigTpv.FechaOffLine=Hoy
            ConfigTpv.HoraOffLine=Hoy
            ConfigTpv.ModoTrabajo=Off-Line
            Modificar registro ConfigTpv
 Sino
    TablaCliente.ErrorEnvio=Verdadero
    TablaCliente.EnvioCentral=Falso
    Modificar registro TablaCliente
```

Código Original

```
Conf.RESET;
Conf.SETFILTER(Conf."Cód. TPV",numTpv);
IF NOT Conf.FIND('-') THEN
  ERROR(Text00001)
ELSE
BEGIN
 IF Conf."Modo de Trabajo"= Conf."Modo de Trabajo"::"On-Line" THEN
 BEGIN
   IF WS.TPV_Alta_Clientes(Conf."Cód. tienda",Conf.Password,
                      "No.",Name,Address,"Post Code",City,
                      County,"Phone No.","Telex No.","E-Mail",
                      "VAT Registration No.","Payment Method Code",
                      "Salesperson Code","Payment Terms Code",
                      "Customer Posting Group","Gen. Bus. Posting Group",
                      "VAT Bus. Posting Group") THEN
   BEGIN
    "Error envío":=FALSE;
    "Enviado a CENTRAL":=TRUE;
    MODIFY;
   END
   ELSE
   BEGIN
    "Error envío":=TRUE;
    "Enviado a CENTRAL":=FALSE;
    MODIFY;
    Conf."Nº de fallos":=Conf."Nº de fallos"+1;
    Conf.MODIFY;
    IF Conf."Nº de fallos" > 3 THEN
    BEGIN
```

```
        Conf."Fecha entrada Off-Line":=WORKDATE;
        Conf."Hora Entrada Off-Line":=TIME;
        Conf."Modo de Trabajo":=Conf."Modo de Trabajo"::"Off-Line";
        Conf.MODIFY;
      END;
    END;
  END
  ELSE
  BEGIN
    "Error envío":=TRUE;
    "Enviado a CENTRAL":=FALSE;
    MODIFY;
  END;
END;
```

Transferencia a Central

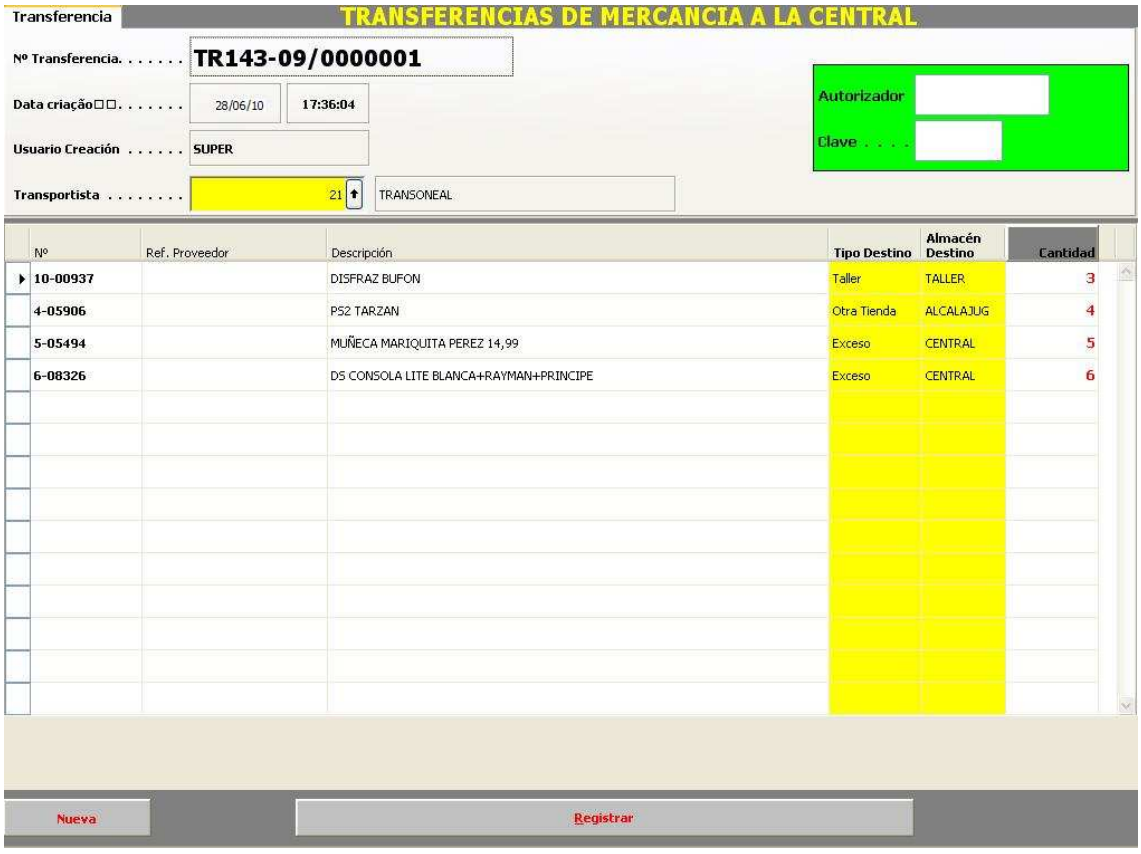


Ilustración 73 Formulario de Transferencia a Central

Los encargados de tienda tienen la posibilidad de mandar mercancía a la Central para que sea tratada según se haya indicado. Los motivos pueden ser varios:

Exceso: En este caso, se puede deber a que el producto se ha descatalogado y se quiere devolver a central para que cuando sea el momento se pueda volver a enviar o que el

producto no haya tenido éxito y en la tienda no es necesario tener tal cantidad de producto.

Taller: En este caso, el producto necesita ser reparado y se debe enviar a central para que el personal de taller puede inspeccionar el producto y ver la causa que ha producido la avería.

Otra Tienda: En este caso, una tienda necesita con urgencia un producto que un cliente ha pedido y en almacén carecen de dicho producto. Entonces, si una tienda tiene ese producto, tiene la posibilidad de mandarlo a Central para que posteriormente se mande a la tienda destino.

Rotura: En caso de que el producto haya llegado a la tienda deteriorado, se debe mandar a CENTRAL indicando que es una rotura para que sea tratado según proceda.

Proveedor: En el caso de que se haya negociado con el proveedor la devolución de la totalidad de un producto, la tienda debe indicar que el producto es devolución del proveedor para su posterior manipulación en almacén.

Como se muestra en la ilustración 73, lo que debe hacer el encargado de la tienda es insertar todos los productos que quiere mandar, la cantidad y el motivo. Una vez hecho ese proceso, debe registrar la transferencia.

En caso de que uno de los motivos sea Exceso, en el formulario salta la parte de autorizador y clave, ya que para estos casos alguien ha debido autorizar la devolución de esos productos. En caso de que los datos introducidos sean correctos, el sistema permite registrar la transferencia.

Registrar(CabTransfer)

PseudoCódigo

```
Si CabTransfer.Enviada entonces
    ERROR('Transferencia ya enviada');
Si NO Confirmas(¿Deseas registrar Transferencia?,Falso) entonces
    ERROR('Proceso Cancelado');
AlmacenVacio=Falso
TipoDestino=Falso
SinCantidad=Falso
FaltanImprimir=Falso
LinTransfer se inicializa
LinTransfer se filtra NoDocumento sea CabTransfer.No
Si encuentra registros LinTransfer entonces
  Repetir
     Si LinTransfer.TipoDestino='' entonces
        TipoDestino=Verdadero
     Si LinTransfer.AlmacenDestino='' entonces
```

```
        AlmacenVacio=Verdadero
      Si LinTransfer.Cantidad <= 0 entonces
        SinCantidad=Verdadero
      Si LinTransfer.TipoDestino=Exceso entonces
        TieneExceso=Verdadero
      Si LinTransfer.Cantidad > LinTransfer.NoBultosImpresos entonces
        FaltanImprimir=Verdadero
      Prod=LinTransfer.No
      Linea=LinTransfer.NoLinea
      Descripción=LinTransfer.Descripcion
      NoBultosImpresos= LinTransfer.NoBultosImpresos
      Cantidad= LinTransfer.Cantidad
  Hasta AlmacenVacio O TipoDestino O SinCantidad O FaltaImprimir O
        Siguiente registro LinTransfer sea vacío
  Si TipoDestino entonces
     ERROR('La línea %1, producto %2 no tiene especificado Tipo Destino, linea,
           Producto)
  Si AlmacenVacio entonces
     ERROR('La línea %1, producto %2 no tiene especificado Almacén Destino,
           Producto)
  Si SinCantidad entonces
     ERROR('La línea %1, producto %2  debe tener una cantidad, producto)
  Si FaltanImprimir entonces
    ERROR('Del producto %1 \       se han impreso %2 etiquetas para %3 bulto(s)...,
          \ Tendrá que  imprimirlas y pegarlas para enviar todo \       por : %4,
          descripción, NoBultosImpresos, Cantidad, Transportista)
Sino
   ERROR('No existen líneas que registrar');

Si TieneExceso Y NO(CabTransfer.Autorizada) entonces
    Activar Autorizador
   Activar Contraseña
   ERROR('Para enviar transferencias por exceso debes validar la auotrización
         Correspondiente);
ConfigTpv se inicializa
Si no encuentra primer registro ConfigTpv entonces
   ERROR('No existe configuración tienda.');
AuxCabTransfer se inicializa
AuxCabTransfer se filtra No sea CabTransfer.No
Si no encuentra registro AuxCabTransfer entonces
   ERROR('Error grave, no existe cab. reposición %1.',CabTransfer.No);
AuxCabTransfer.FechRegistro=FechaTrabajo
AuxCabTransfer.FechRegistroTienda=FechaHoraTrabajo
AuxCabTransfer.UsuarioRegistro=USERID
AuxCabTransfer.FechaEnvio=Hoy
Modificar Registro AuxCabTransfer

PasarHistorico(AuxCabTransfer)
HistoricoTransfer se inicializa
HistoricoTransfer.Coger(CabTransfer.No);
Borrar AuxCabTransfer;

AuxCabTransfer se inicializa
AuxCabTransfer.Coger(CabTransfer.No);
```

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea Ntpv
ConfigTpv se posiciona en el primer registro
ImprimirTransferencia(AuxCabTransfer);
Si ServicioWeb.TPV_Transfer_Central(ConfigTPV.CodTienda, ConfigTpv.Password,
    HistoricoTransfer.No);
    HistoricoTransfer.Enviada=Verdadero
    HistoricoTransfer.ErrorEnvio=Falso
    Modificar registro HistoricoTransfer
    Message('Transferencia Enviada correctamente');
```

Código Original

```
IF Enviada THEN
  ERROR(Text00006);

IF NOT CONFIRM(Text00007,FALSE) THEN
   ERROR(Text00008);

AlmacenVacio:=FALSE;
TipoDestVacio:=FALSE;
SinCantidad:=FALSE;
FaltanBultosxImprimir:=FALSE;
TESTFIELD(Rec.Tansportista);

RcdLinReposicion.RESET;
RcdLinReposicion.SETFILTER("Nº documento",Rec."Nº");
IF RcdLinReposicion.FINDSET(FALSE,FALSE) THEN BEGIN
  REPEAT
    IF RcdLinReposicion."Tipo Destino"=RcdLinReposicion."Tipo Destino"::" " THEN
      TipoDestVacio:=TRUE;
    IF RcdLinReposicion."Almacen Destino"=" THEN
      AlmacenVacio:=TRUE;
    IF RcdLinReposicion.Cantidad <= 0 THEN
      SinCantidad:=TRUE;

    IF RcdLinReposicion."Tipo Destino"= RcdLinReposicion."Tipo Destino"::Exceso THEN
      bTieneExceso:=TRUE;

    IF RcdLinReposicion.Cantidad > RcdLinReposicion."No Bultos Impresos" THEN
      FaltanBultosxImprimir:=TRUE;
    MProd:=RcdLinReposicion."Nº";
    Mlinea:=RcdLinReposicion."Nº línea";
    MDescripcion:=RcdLinReposicion.Descripción;
    MNoBultosImpresos:=RcdLinReposicion."No Bultos Impresos";
    MCantidad:=RcdLinReposicion.Cantidad;
UNTIL(AlmacenVacio)OR(TipoDestVacio)OR(SinCantidad)OR(FaltanBultosxImprimir)OR(R
cdLinReposicion.NEXT=0);

  IF TipoDestVacio THEN
    ERROR(Txt0002,Mlinea,MProd);
  IF AlmacenVacio THEN
    ERROR(Txt0003,MProd);
  IF SinCantidad THEN
```

```
    ERROR(Txt0004,MProd);

  IF FaltanBultosxImprimir THEN
    ERROR(tXT0005,MDescripcion,MNoBultosImpresos,MCantidad,txtTransportista);
END ELSE
  ERROR(Txt0001);

IF (bTieneExceso) AND (NOT Autorizada) THEN
BEGIN
  CurrForm.cntAutoriz.VISIBLE(TRUE);
  CurrForm.lblAutorizador.VISIBLE(TRUE);
  CurrForm.cntAutorizador.VISIBLE(TRUE);
  CurrForm.lblClave.VISIBLE(TRUE);
  CurrForm.cntPwd.VISIBLE(TRUE);
  ERROR(Text00009);

END;

ConfTPV.RESET;
IF NOT ConfTPV.FINDFIRST THEN
 ERROR(Text00010);

CabRep.RESET;
CabRep.SETRANGE(CabRep."Nº", "Nº");
IF NOT CabRep.FINDFIRST THEN
   ERROR(Text00011,"Nº");

CabRep."Fecha registro":=WORKDATE;
CabRep."Fecha Registro Tienda":=CURRENTDATETIME;
CabRep."Usuario Registro":=USERID;
CabRep."Fecha envío":=TODAY;
CabRep.MODIFY;

cduILR.PasarTransAHistórico(CabRep);

rcdHco.RESET;
IF rcdHco.GET("Nº") THEN;

CabRep.DELETE(TRUE);
COMMIT;
rcdTrans.SETRANGE("Nº","Nº");
IF rcdTrans.FINDFIRST THEN;

Ntpv:=cduILR.DameTPV();
Config.RESET;
Config.SETFILTER("Cód. TPV", Ntpv);
Config.FINDFIRST;
CLEAR(rptTransNegra);
rptTransNegra.SETTABLEVIEW(rcdTrans);
rptTransNegra.USEREQUESTFORM(FALSE);
rptTransNegra.RUNMODAL;
```

```
IF      WS.TPV_Transfer_Central(ConfTPV."Cód.      tienda",ConfTPV.Password,rcdHco."Nº") THEN
BEGIN
  rcdHco.Enviada := TRUE;
  rcdHco."Error envío":=FALSE;
  rcdHco.MODIFY;
  MESSAGE('Transferencia %1 enviada correctamente',rcdHco."Nº");
END;
```

Validar

PseudoCódigo

```
Si Autorizador=’’ entonces
   ERROR(‘Debes indicar el código de autorizador’);
Si Password=’’ entonces
   ERROR(‘Debes indicar la clave de autorización’);
Si No ServicioWeb.TPV_Autorizacion_Exceso(Autorizador,Password,Nombre,Fecha)
   ERROR(‘No existe ninguna autorización activa por %1.\+
            ‘Verifica que el autorizador y su clave sean correctos’,Autorizador);
Si Fecha<Hoy entonces
   ERROR(‘La Autorización Caducó el %1,Fecha);
Message(‘Transferencia autorizada por %1,Nombre);
CabTransfer.Autorizador=Autorizador
CabTransfer.ClaveAutorizacion=Password
Modificar registro CabTransfer
```

Código Original

```
IF optAutorizador = '' THEN
  ERROR(Text00001);
IF optPwd='' THEN
  ERROR(Text00002);
IF                                                                    NOT
WS.TPV_Autorizacion_Exceso(optAutorizador,optPwd,PARNombre,PARFechaCaduc) THEN
  ERROR(Text00003+Text00004,optAutorizador);

IF PARFechaCaduc < TODAY THEN
    ERROR(Text00005,PARFechaCaduc);
MESSAGE(Text00006,PARNombre);
Autorizador:=optAutorizador;
"Clave Autorizacion":=optPwd;
Autorizada:=TRUE;
MODIFY;
```

TRANSFERENCIA A CENTRAL

Copia Transportista

Nº: **TR143-10/0000008**

Origen **LAGAVIA Vallecas**

Fec.Crea: 14/05/10 16:04:00

Fec.Reg: 17/01/11 17:58

Ref.	Descripción	Cant.
	Otra Tienda	
10-00283	OSO CORAZON 75 CM	2
5-07926	PS2 DRAGON BALL BUDOKA	1
5-08981	Coleccion Coches Esc. 1/18	3

Nº Total de Artículos:6

SUPER

Firma Encargado

3 - CORREOS

Firma Transportista

Ilustración 74 Impresión al registrar la transferencia

Transferencia a Coordinador

Este tipo de transferencia es parecida al tipo “otra tienda” en las transferencias a central pero en este caso existe un almacén de tránsito de la mercancía, que será el almacén asignado al coordinador que se quiere llevar el producto de una tienda a otra.

El proceso es análogo al de transferencia a central. En este caso, el coordinador pone los productos que se quiere llevar y la cantidad. Una vez indicado cuales son, debe introducir su nombre y contraseña para poder registrar la transferencia. Entonces, el sistema descuenta de la tienda esos productos y los carga en el almacén del coordinador para que los pueda dejar en la tienda que los necesite.

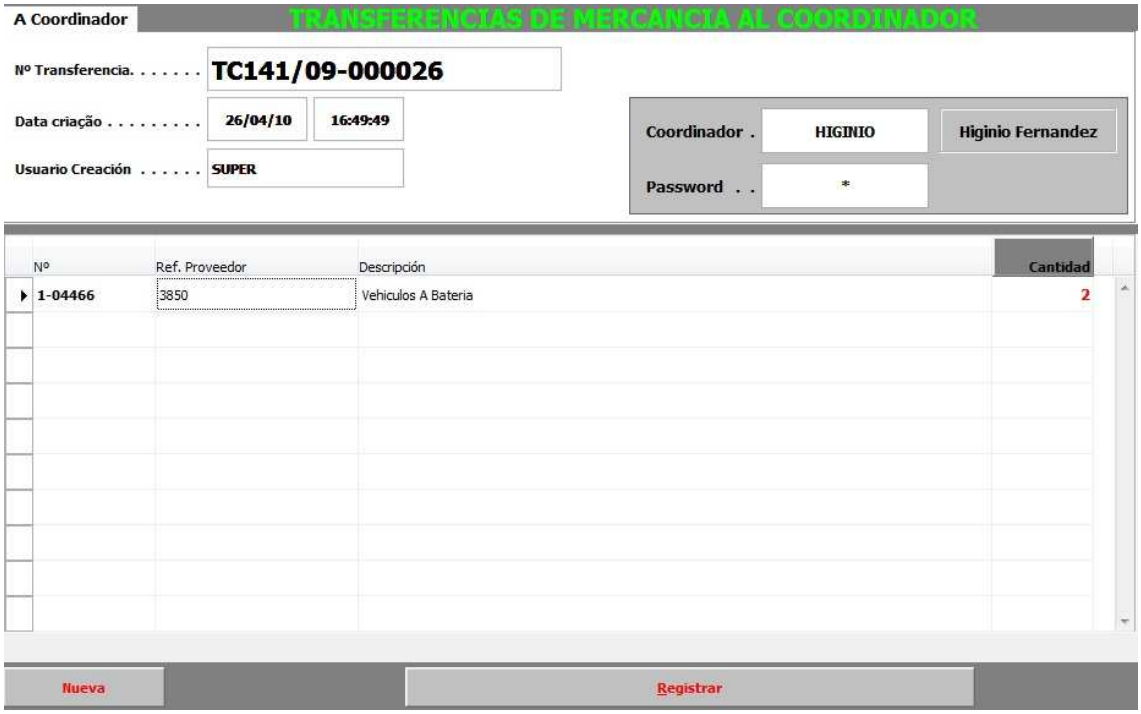


Ilustración 75 Formulario de Transferencia a Coordinador

Validar

PseudoCódigo

```
Si Coordinador=’’ entonces
   ERROR(‘Debes indicar el código del coordinador’);
Si Password=’’ entonces
   ERROR(‘Debes indicar la contraseña’);
Si NO ServicioWeb.TPV_Login_Coordinador(Coordinador, Password, Falso,
        NombreCoordinador) entonces
   ERROR(‘Usuario o Password incorrectos’);
Habilitar Boton registrar.
```

Código Original

```
IF optCoordinador = '' THEN
  ERROR(Text00001);

IF optPwd='' THEN
  ERROR(Text00002);

txtnomCoordinador:='';
CLEAR(WSLocal);
```

```
IF NOT
WSLocal.TPV_Login_Coordinador(optCoordinador,optPwd,FALSE,txtnomCoordinador)
THEN
  ERROR(Text00003):

CurrForm.txtNombre.UPDATE();
CurrForm.cntCoordinador.EDITABLE(FALSE);
CurrForm.cntPwd.EDITABLE(FALSE);
CurrForm.LineasVenta.EDITABLE(FALSE);
CurrForm.cntRegistrar.VISIBLE(TRUE);
CurrForm.cmdValidar.VISIBLE(FALSE);
```

Registrar(CabTransfer)

PseudoCódigo

```
Si CabTransfer.Enviada entonces
   ERROR('La transferencia ya está enviada.');
Si NO Confirma('¿Estás seguro de entregar esta mercancía a el Coordinador'+
        NombreCoordinador+'?',FALSO) entonces
   ERROR('Proceso cancelado');
LinTransfer se inicializa
LinTransfer se filtra NoDocumento sea CabTransfer.No
Si encuentra registros LinTransfer entonces
  Repetir
     LinTransfer.AlmacenDestino=Coordinador
     Modificar registro LinTransfer
  Hasta siguiente registro LinTransfer sea vacío
SinCantidad=Falso
LinTransfer se inicializa
LinTransfer se filtra NoDocumento sea CabTransfer.No
Si encuentra registros LinTransfer entonces
   Repetir
      Si LinTransfer.Cantidad<=0 entonces
          SinCantidad=Verdadero
   Hasta (siguiente registro LinTransfer sea vacio) O (SinCantidad)
   Si SinCantidad entonces
      ERROR('La línea %1 debe tener una cantidad', LinTransfer.NoLinea)
Sino
   ERROR('No existen líneas que registrar');

ConfigTpv se inicializa
Si No se posiciona primer registro ConfigTpv entonces
   ERROR('No existe configuración tienda.');
AuxCabTransfer se inicializa
AuxCabTransfer se filtra No sea CabTransfer.No
Si no encuentra registro AuxCabTransfer entonces
   ERROR('Error grave, no existe cab. reposición %1.',CabTransfer.No)
AuxCabTransfer.FechaRegistro=Hoy;
AuxCabTransfer.FechaRegistroTienda=Hoy;
AuxCabTransfer.Destino=Coordinador;
AuxCabTransfer.DescripcionDestino=NombreCoordinador;
AuxCabTransfer.FechaEnvio=Hoy;
Modificar registro AuxCabTransfer;
```

```
Pasar Transferencia a Historico;
HistTranfer se inicializa
Si HisTransfer.Coge(CabTransfer.No) entonces;
Borrar registro AuxCabTransfer;

AuxCabTransfer se filtra No sea CabTransfer.No
Si encuentra registro AuxCabTransfer entonces;
ImprimirTransferencia;

Si ServicioWeb.Tpv_Transfer_Coordinador(ConfigTpv.CodTienda,
   ConfigTpv.Password,HistTransfer.No) entonces
   HistTransfer.Enviada=Verdadero;
   HistTransfer.ErrorEnvio=Falso;
   Modificar registro HistTransfer;
  Message('Transferencia a coordinador %1 enviada correctamente a CENTRAL',
            HistTransfer.No);
```

Código Original

```
IF Enviada THEN
  ERROR(Text00005);

IF NOT CONFIRM(Text00006+FORMAT(txtnomCoordinador)+'?',FALSE) THEN
   ERROR(Text00007);
RcdLinReposicion.RESET;
RcdLinReposicion.SETFILTER("Nº documento",Rec."Nº");
IF RcdLinReposicion.FINDSET(FALSE,FALSE) THEN
BEGIN
   REPEAT
     RcdLinReposicion."Almacen Destino":=optCoordinador;
     RcdLinReposicion.MODIFY;
   UNTIL (RcdLinReposicion.NEXT=0);
END;
//-002
SinCantidad:=FALSE;
RcdLinReposicion.RESET;
RcdLinReposicion.SETFILTER("Nº documento",Rec."Nº");
IF RcdLinReposicion.FINDSET(FALSE,FALSE) THEN BEGIN
   REPEAT
      IF RcdLinReposicion.Cantidad <= 0 THEN
        SinCantidad:=TRUE;
UNTIL(RcdLinReposicion.NEXT=0)OR(AlmacenVacio)OR(TipoDestVacio)OR(SinCantidad);
   IF SinCantidad THEN
      ERROR(Txt0004,RcdLinReposicion."Nº línea");
END ELSE
   ERROR(Txt0001);
//+002
ConfTPV.RESET;
IF NOT ConfTPV.FINDFIRST THEN
  ERROR(Text00008);
CabRep.RESET;
CabRep.SETRANGE(CabRep."Nº", "Nº");
IF NOT CabRep.FINDFIRST THEN
    ERROR(Text00009,"Nº");
```

```
CabRep."Fecha registro":=WORKDATE;
CabRep."Fecha Registro Tienda":=CURRENTDATETIME;
CabRep.Destino:=optCoordinador;
CabRep."Descripcion Destino":=txtnomCoordinador;
CabRep."Fecha envío":=TODAY;
CabRep.MODIFY;
cduILR.PasarTransAHistórico(CabRep);

rcdHco.RESET;
IF rcdHco.GET("Nº") THEN;

CabRep.DELETE(TRUE);
COMMIT;



rcdTrans.SETRANGE("Nº","Nº");
IF rcdTrans.FINDFIRST THEN;

// Vemos la impresora que tiene habilitada este tpv
Ntpv:=cduILR.DameTPV();
CLEAR(rptTransNegra);
rptTransNegra.SETTABLEVIEW(rcdTrans);
rptTransNegra.USEREQUESTFORM(FALSE);
rptTransNegra.RUNMODAL;

IF  WS.TPV_Transfer_Coordinador(ConfTPV."Cód.  tienda",ConfTPV.Password,rcdHco."Nº")
THEN
BEGIN
  rcdHco.Enviada := TRUE;
  rcdHco."Error envío":=FALSE;
  rcdHco.MODIFY;

  MESSAGE(Text00010,rcdHco."Nº");
END;
CurrForm.CLOSE;
```

TRANSFERENCIA A COORDINADORES

Copia Coordinador

TC141/09-000026

Nº: **TC141/09-000026**

Origen **LAGAVIA** **Vallecas**

Destino **HIGINIO** **Higinio Fernande**

Fec.Crea: 26/04/10 16:49:49

Fec.Reg: 19/01/11 17:00

Ref.	Descripción	Cant.
1-04466	Vehiculos A Bateria	2
9-08368	FORCEVALOR TANQUE AL	2

Nº Total de Artículos:4

SUPER

Firma Encargado

Higinio Fernandez

Firma Coordinador

Ilustración 76 Ejemplo de impresión al registrar transferencia

Recibir de Coordinador

Este apartado corresponde a la segunda función que debe realizar el coordinador, es decir, que debe dejar los productos, que cogió en la tienda origen, en la tienda donde los necesita para una venta.

Ilustración 77 Primera Pantalla que sale al acceder al formulario

Ilustración 78 Segunda Pantalla que sale al validar el usuario en el formulario

En primer lugar, el coordinador debe introducir su nombre y su contraseña y validar que lo introducido es correcto, como se muestra en la ilustración 77. Una vez validados correctamente, se debe introducir el número de transferencia de la cual se quieren dejar los productos.

Entonces, el sistema se conecta a la CENTRAL y comprueba si existe la transferencia. En caso de existir, como muestra la ilustración 78, el sistema muestra los productos que están incluidos en la transferencia. Por lo tanto, el coordinador en la columna que está en amarillo, debe indicar qué cantidad de cada producto quiere dejar en la tienda.

Una vez se haya indicado la cantidad, el coordinador registra la transferencia para que el sistema pueda generar los movimientos correspondientes.

Validar

PseudoCódigo

```
Si Coordinador='' entonces
    ERROR('Debes indicar el código del coordinador');
Si Password='' entonces
```

```
    ERROR('Debes indicar la contraseña');
Si NO ServicioWeb.Tpv_Login_Coordinador(Coordinador, Password, Falso,
        NombreCoordinador) entonces
   ERROR('Usuario o Password incorrectos');
Habilitar Etiqueta Transferencia
```

Código Original

```
IF optCoordinador = '' THEN
  ERROR(Text00001);

IF optPwd='' THEN
  ERROR(Text00002);

IF NOT WS.TPV_Login_Coordinador(optCoordinador,optPwd,FALSE,txtnomCoordinador)
THEN
  ERROR(Text00003);

CurrForm.lblTransfer.VISIBLE(TRUE);
CurrForm.cntTransfer.VISIBLE(TRUE);
CurrForm.cntCoordinador.EDITABLE(FALSE);
CurrForm.cntPwd.EDITABLE(FALSE);
CurrForm.cntTransfer.ACTIVATE;
```

ValidarTransferencia(Numero)

PseudoCódigo

```
Si ServicioWeb.Tpv_Coordinador_BuscarTransfer(Numero,Coordinador)=0 entonces
   ERROR('No existe información del ticket');
```

Código Original

```
IF WS.TPV_Coordinador_BuscaTransfer(optTransferencia,optCoordinador) = 0 THEN
  ERROR(Text00001);
```

Registrar(CabTransfer)

PseudoCódigo

```
LinTransfer se inicializa
LinTransfer se filtra NoDocumento sea CabTransfer.No
LinTransfer se filtra CantidadRecibida sea mayor que 0
Si no encuentra registros LinTransfer entonces
  ERROR('No hay nada marcado para entregar');

Repetir
   Artículos:=Artículos+1;
   Cantidad:=Cantidad+LinTransfer.CantidadRecibida
Hasta siguiente registro LinTransfer sea vacío
Si NO Confirma(¿Estás seguro que deseas entregar %1 artículos y %2 unidades en esta
        tienda?,FALSO,Articulos,Cantidad);
   ERROR('Proceso cancelado');
ConfigTpv se inicializa
```

```
Si NO encuentra primer registro ConfigTpv entonces
   ERROR(‘No existe configuración tienda.’);
AuxCabTransfer se inicializa
AuxCabTransfer se filtra No sea CabTransfer.No
Si no encuentra registros AuxCabTransfer entonces
   ERROR(‘Error grave, no existe cab. reposición %1., AuxCabTransfer.No);
Numero:=ServicioWeb.TPV_Coordinador_Numero(AuxCabTransfer.No);
HistTransfer se inicializa
HistTransfer.TransferirCampos(AuxCabTransfer);
HisTransfer.No=Numero;
HistTransfer.FechaRegistroTienda=Hoy
HistTransfer.FechaRegistro=Hoy
HistTransfer.ErrorEnvio=Verdadero
Insertar registro HistTransfer
LinTransfer se inicializa
LinTransfer se filtra NoDocumento sea AuxCabTransfer.No
LinTransfer se filtra CantidadRecibida mayor que 0
Si encuentra registros LinTransfer entonces
  Repetir
    HistLinTransfer se inicializa
    HistLinTransfer.NoDocumento=Numero;
    HistLinTransfer.NoLinea=LinTransfer.NoLinea
    HistLinTransfer.No=LinTransfer.No
    HistLinTransfer.Descripcion=LinTransfer.Descripcion
    HistLinTransfer.Cantidad=LinTransfer.CantidadRecibida
Hasta siguiente registro LinTransfer sea vacio
Borrar registro AuxCabTransfer
ImprimirTransferencia;
Si ServicioWeb.TPV_Entrega_Coordinador (ConfigTpv.CodTienda,
     ConfigTpv.Password, HistTransfer.No) entonces
     HistTransfer.Enviada=Verdadero;
     HistTransfer.ErrorEnvio=Falso;
     Modificar registro HistTransfer
     Message (‘Entrega de coordinador %1 enviada correctamente, HistTransfer.No);
```

Código Original

```
RcdLinReposicion.RESET;
RcdLinReposicion.SETFILTER("Nº documento",AuxCabRep."Nº");
RcdLinReposicion.SETFILTER("Cantidad recibida",'>0');
IF NOT RcdLinReposicion.FINDSET(FALSE,FALSE) THEN
  ERROR(Text00005);

REPEAT
  nvArticulos+=1;
  nvCantidad+=RcdLinReposicion."Cantidad recibida";
UNTIL (RcdLinReposicion.NEXT=0);
IF NOT CONFIRM(Text00006,FALSE,FORMAT(nvArticulos),FORMAT(nvCantidad))
THEN
  ERROR(Text00007);

ConfTPV.RESET;
IF NOT ConfTPV.FINDFIRST THEN
  ERROR(Text00008);
CabRep.RESET;
```

```
CabRep.SETRANGE(CabRep."Nº", AuxCabRep."Nº");
IF NOT CabRep.FINDFIRST THEN
   ERROR(Text00009,AuxCabRep."Nº");

cvTransf:=WS.TPV_Coordinador_NumTransfer(CabRep."Nº");
rcdHco.INIT;
rcdHco.TRANSFERFIELDS(CabRep);
rcdHco."Nº":=cvTransf;
rcdHco."Fecha Registro Tienda":=CURRENTDATETIME;
rcdHco."Fecha registro":=TODAY;
rcdHco."Error envío":=TRUE;
rcdHco.INSERT;
RcdLinReposicion.RESET;
RcdLinReposicion.SETFILTER("Nº documento",AuxCabRep."Nº");
RcdLinReposicion.SETFILTER("Cantidad recibida",'>0');
IF RcdLinReposicion.FINDSET(FALSE,FALSE) THEN
REPEAT
  RcdHcoLinreposicion.INIT;
  RcdHcoLinreposicion."Nº documento":=cvTransf;
  RcdHcoLinreposicion."Nº línea":=RcdLinReposicion."Nº línea";
  RcdHcoLinreposicion."Nº":=RcdLinReposicion."Nº";
  RcdHcoLinreposicion.Descripción:=RcdLinReposicion.Descripción;
  RcdHcoLinreposicion.Cantidad:=RcdLinReposicion."Cantidad recibida";
  RcdHcoLinreposicion.INSERT;
UNTIL RcdLinReposicion.NEXT=0;
CabRep.DELETE(TRUE);
rcdHco.SETRANGE("Nº",cvTransf);
IF rcdHco.FINDFIRST THEN;
rptTransNegra.SETTABLEVIEW(rcdHco);
rptTransNegra.USEREQUESTFORM(FALSE);
rptTransNegra.RUNMODAL;

IF WS.TPV_Entrega_Coordinador(ConfTPV."Cód. tienda",ConfTPV.Password,rcdHco."Nº")
THEN
BEGIN
 rcdHco.Enviada := TRUE;
 rcdHco."Error envío":=FALSE;
 rcdHco.MODIFY;
 MESSAGE(Text00010,rcdHco."Nº");
END;
```

ENTREGAS DE COORDINADOR A TIENDA

Copia Coordinador

Nº: **TC141/09-000026/001**

Origen **HIGINIO**

Destino **LAGAVIA**

Fec.Reg: 20/01/11 11:48

Ref.	Descripción	Cant.
1-04466	FAMOSA MOTO MOVISTAR	1

Nº Total de Artículos: 1

Firma Encargado

Firma Coordinador

Ilustración 79 Ejemplo de impresión al registrar la transferencia

Robos

Este tipo de transferencia es análoga a la de transferir a coordinador, debido a que los pasos a seguir son idénticos: en primer lugar se deben indicar los productos que se van a transferir, después se debe validar el usuario y por último, se debe registrar la transferencia. La única diferencia es que a la hora del registro, en este caso la mercancía será cargada en el almacén ROBOS.

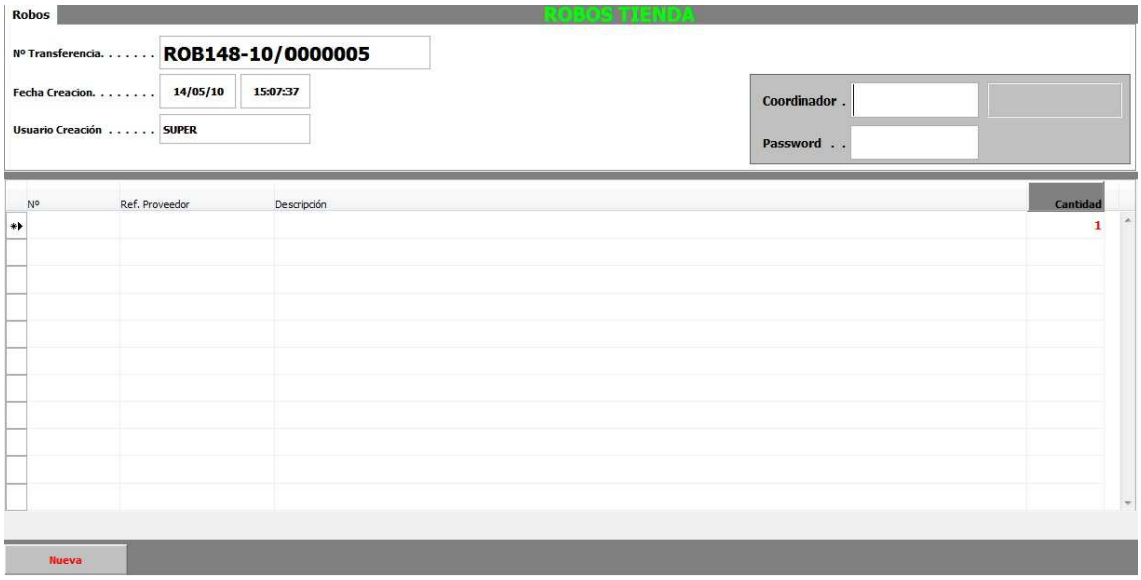


Ilustración 80 Primera Pantalla que sale al acceder el formulario

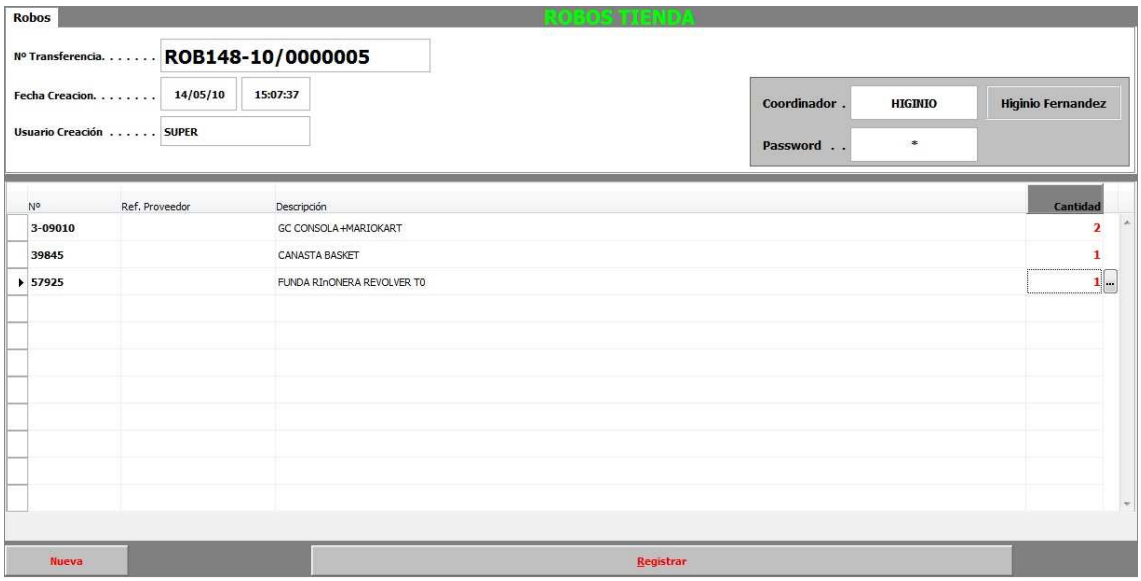


Ilustración 81 Segunda Pantalla que sale al validar el usuario

Como muestran las ilustraciones 80 y 81, se puede observar que tanto la transferencia a coordinador como a robos son parecidas, lo único que cambia a qué almacén se carga el stock de los productos.

Validar

PseudoCódigo

```
Si Coordinador='' entonces
    ERROR ('Debes indicar el código del coordinador');
Si Password='' entonces
   ERROR ('Debes indicar la contraseña');
Si NO ServicioWeb.Tpv_Login_Coordinador(Coordinador,Password,Falso,
        NombreCoordinador) entonces
   ERROR('Usuario o Password incorrectos');
```

Código Original

```
IF optCoordinador = '' THEN
  ERROR(Text00001);

IF optPwd='' THEN
  ERROR(Text00002);

CLEAR(WS);
IF NOT WS.TPV_Login_Coordinador(optCoordinador,optPwd,FALSE,txtnomCoordinador)
THEN
  ERROR(Text00003);

CurrForm.cntNombre.UPDATE();
CurrForm.cntCoordinador.EDITABLE(FALSE);
CurrForm.cntPwd.EDITABLE(FALSE);
CurrForm.LineasVenta.EDITABLE(FALSE);
CurrForm.cntRegistrar.VISIBLE(TRUE);
CurrForm.cmdValidar.VISIBLE(FALSE);
```

Registrar(CabTransfer)

PseudoCódigo

```
Si CabTransfer.Enviada entonces
   ERROR('La transferencia ya está enviada.');
Si NO Confirma('¿Estás seguro de entregar esta mercancía a el Coordinador'+
        NombreCoordinador+'?',FALSO) entonces
   ERROR('Proceso cancelado');
LinTransfer se inicializa
LinTransfer se filtra NoDocumento sea CabTransfer.No
Si encuentra registros LinTransfer entonces
  Repetir
     LinTransfer.AlmacenDestino=Coordinador
     Modificar registro LinTransfer
  Hasta siguiente registro LinTransfer sea vacío
SinCantidad=Falso
LinTransfer se inicializa
LinTransfer se filtra NoDocumento sea CabTransfer.No
Si encuentra registros LinTransfer entonces
   Repetir
      Si LinTransfer.Cantidad<=0 entonces
         SinCantidad=Verdadero
   Hasta (siguiente registro LinTransfer sea vacio) O (SinCantidad)
   Si SinCantidad entonces
      ERROR('La línea %1 debe tener una cantidad', LinTransfer.NoLinea)
Sino
   ERROR('No existen líneas que registrar');

ConfigTpv se inicializa
Si No se posiciona primer registro ConfigTpv entonces
   ERROR('No existe configuración tienda.');
AuxCabTransfer se inicializa
AuxCabTransfer se filtra No sea CabTransfer.No
Si no encuentra registro AuxCabTransfer entonces
```

```
    ERROR(‘Error grave, no existe cab. reposición %1.’,CabTransfer.No)
AuxCabTransfer.FechaRegistro=Hoy;
AuxCabTransfer.FechaRegistroTienda=Hoy;
AuxCabTransfer.Destino=Coordinador;
AuxCabTransfer.DescripcionDestino=NombreCoordinador;
AuxCabTransfer.FechaEnvio=Hoy;
Modificar registro AuxCabTransfer;
Pasar Transferencia a Historico;
HistTranfer se inicializa
Si HisTransfer.Coge(CabTransfer.No) entonces;
Borrar registro AuxCabTransfer;

AuxCabTransfer se filtra No sea CabTransfer.No
Si encuentra registro AuxCabTransfer entonces;
ImprimirTransferencia;

Si ServicioWeb.Tpv_Transfer_Robos(ConfigTpv.CodTienda,
    ConfigTpv.Password,HistTransfer.No) entonces
    HistTransfer.Enviada=Verdadero;
    HistTransfer.ErrorEnvio=Falso;
    Modificar registro HistTransfer;
   Message(‘Transferencia a coordinador %1 enviada correctamente a CENTRAL’,
                HistTransfer.No);
```

Código Original

```
IF Enviada THEN
  ERROR(Text00005);

IF NOT CONFIRM(Text00006+FORMAT(' ROBOS')+'?',FALSE) THEN
   ERROR(Text00007);

SinCantidad:=FALSE;
RcdLinReposicion.RESET;
RcdLinReposicion.SETFILTER("Nº documento",Rec."Nº");
IF RcdLinReposicion.FINDSET(FALSE,FALSE) THEN BEGIN
   REPEAT
     IF RcdLinReposicion.Cantidad <= 0 THEN
       SinCantidad:=TRUE;
UNTIL(RcdLinReposicion.NEXT=0)OR(AlmacenVacio)OR(TipoDestVacio)OR(SinCantidad);
   IF SinCantidad THEN
     ERROR(Txt0004,RcdLinReposicion."Nº línea");
END ELSE
   ERROR(Txt0001);

ConfTPV.RESET;
IF NOT ConfTPV.FINDFIRST THEN
  ERROR(Text00008);

CabRep.RESET;
CabRep.SETRANGE(CabRep."Nº", "Nº");
IF NOT CabRep.FINDFIRST THEN
    ERROR(Text00009,"Nº");

CabRep."Fecha registro":=WORKDATE;
CabRep."Fecha Registro Tienda":=CURRENTDATETIME;
```

```
CabRep.Destino:='ROBOS';
CabRep."Descripcion Destino":=optCoordinador;
CabRep."Fecha envío":=TODAY;
CabRep.MODIFY;

cduILR.PasarTransAHistórico(CabRep);

rcdHco.RESET;
IF rcdHco.GET("Nº") THEN;

CabRep.DELETE(TRUE);
rcdTrans.SETRANGE("Nº","Nº");
IF rcdTrans.FINDFIRST THEN;

Ntpv:=cduILR.DameTPV();
CLEAR(rptTransNegra);
rptTransNegra.SETTABLEVIEW(rcdTrans);
rptTransNegra.USEREQUESTFORM(FALSE);
rptTransNegra.RUNMODAL;

IF WS.TPV_Transfer_Robos(ConfTPV."Cód. tienda",ConfTPV.Password,rcdHco."Nº") THEN
BEGIN
  rcdHco.Enviada := TRUE;
  rcdHco."Error envío":=FALSE;
  rcdHco.MODIFY;
  MESSAGE(Text00010,rcdHco."Nº");
END;
CurrForm.CLOSE;
```

ROBOS TIENDA

Copia Coordinador

Nº: ROB148-10/0000005

Origen LAGAVIA Vallecas

Destino ROBOS ROBOS

Fec.Reg: 20/01/11 15:04

Ref.	Descripción	Cant.
39845	CANASTA BASKET	1
57925	FUNDA RInONERA REVOL	1
3-09010	GC CONSOLA+MARIOKAR	2

Nº Total de Artículos: 4

SUPER

Firma Encargado

HIGINIO

Firma Coordinador

Ilustración 82 Ejemplo de impresión al registrar la transferencia

4.2.6 ServicioWeb

En este apartado se explican las funciones que se invocan en los anteriores apartados y que conciercen al servicio web, que es el encargado de enviar todos los movimientos que se generen en la tienda a CENTRAL, para que puedan procesarse y así poder centralizar los datos. En cada función el objetivo principal es generar un XML, para que el servicio web pueda procesar la petición.

ST_Abrir_Login(Usuario,Password,Error)

Esta función valida que el usuario introducido para enviar la información es válido, ya que de este modo se evita que nadie externo pueda conectarse e ir invocando a las distintas funciones del servicio web.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si EncabezadoWeb.Vacio entonces
    Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/ST_Login');
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXML('<ST_Login xmlns=http://tempuri.org/><userid>'
                +Usuario+'</userid>'+'<Password>'+Password+
                '</Password></ST_Login>')
ConectorWeb.FinCuerpo;
ConectorWeb.FinEstructura;
ConectorWeb.FinMensaje;
Si EncabezadoWeb.Error< >'' entonces
   Limpiar(EncabezadoWeb)
   Si NO Error entonces
      ERROR('ERROR al conectar con la CENTRAL, Inténtelo mas tarde');
   Sino
      Salir(FALSO);
Si NodoXML.vacio entonces
   Crear(NodoXML);
NodoXML.cargar(EncabezadoWeb);
NodoXML.guardar('Respuesta.xml');
NodoLista:=NodoXML.CogerElementoConNombre('Connected');
Si Longitud(NodoLista)=0 entonces
    Salir(Falso);
Conectado:=NodoLista.elemento[0].Texto;
Si Conectado=Verdadero entonces
    Salir(Verdadero)
Sino
    Salir(Falso)
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF Conf.FINDFIRST THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
  END
ELSE
  ERROR('No Existe Configuración TPV');
BEGIN
 IF ISCLEAR(SoapHttpConn) THEN
   CREATE(SoapHttpConn);
 SoapHttpConn.Property('EndPointURL', URLWS);
 SoapHttpConn.Connect;
 SoapHttpConn.Property('SoapAction','http://tempuri.org/ST_Login');
 SoapHttpConn.BeginMessage;
 IF ISCLEAR(SoapSerialize) THEN
   CREATE(SoapSerialize);
 SoapSerialize.Init(SoapHttpConn.InputStream);
 SoapSerialize.StartEnvelope('','STANDARD','');
 SoapSerialize.StartBody('');
 SoapSerialize.WriteXML('<ST_Login           xmlns="http://tempuri.org/"><UserId>'+xUserID
+'</UserId>'+
                                        '<Password>'+ xPassWord+'</Password></ST_Login>');
 SoapSerialize.EndBody;
 SoapSerialize.EndEnvelope;
 SoapHttpConn.EndMessage;
 IF  SoapHttpConn.Error <> '' THEN BEGIN
    CLEAR(SoapHttpConn);
    IF NOT OcultarError THEN
      ERROR(Text0001)
    ELSE
      EXIT(FALSE);
 END;
 IF ISCLEAR(XMLDom) THEN
   CREATE(XMLDom);
 XMLDom.async := FALSE;
 XMLDom.load(SoapHttpConn.OutputStream);
 XMLDom.save(cduILR.DameDirXML() + 'Respuesta.xml');
 DOMNodelist:= XMLDom.getElementsByTagName('Connected');
 IF DOMNodelist.length = 0 THEN
   EXIT(FALSE);
 Conectado:=DOMNodelist.item(0).text;
 IF Conectado='true' THEN BEGIN
   EXIT(TRUE);
 END
 ELSE
  EXIT(FALSE);
END;
```

TPV_Alta_Clientes(Usuario,Password,Codigo,Nombre,Direccion,CodigoPostal, Poblacion,Provincia,Telefono,Telefono2,Email,CIF,FormaPago,Vendedor, TerminosPago,ContableCliente,ContableNegocio,RegistroIVA);

Esta función se utiliza cuando la tienda genera un nuevo cliente en su base de datos. Entonces, el sistema invoca esta función para que dicho cliente sea creado en la base de datos de CENTRAL y se pueden mirar todos los movimientos de saldo que ha ido realizando el cliente en cuestión.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
TablaCliente.Coge(Codigo);
TextoIVA=Formateo(TablaCliente.PrecioIncluidoIVA);
Si EncabezadoWeb.Vacio entonces
    Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_AltaCliente');
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribeXML('<TPV_AltaCliente xmlns=http://tempuri.org/>'+
                '<Usuario>'+Usuario+'</Usuario>'+'<password>'+Password+
                '</password>'+'<Numero>'+Codigo+'</Numero>'+'<Nombre>'+
                Nombre+'</Nombre>'+'<Dirección>'+Direccion+'</Direccion>'+
                '<cp>'+CodigoPostal+'</cp>'+'<población>'+Poblacion+
                '</población>'+'<provincia>'+Provincia+'</provincia>'+
                '<teléfono>'+Telefono+'</teléfono>'+'<telefono2>'+Telefono2+
                '<email>'+Email+'</email>'+'<nif>'+NIF+'</nif>'+'<codformapago>'
               +FormaPago+'</codformapago>'+'<codvendedor>'+Vendedor+
               '</codvendedor>'+'<codterminospago>'+TerminosPago+
               '</codterminospago>'+'<grupocontablecliente>'+ContableCliente+
               '</grupocontablecliente>'+'<grupocontnegocio>'+ContableNegocio+
               '<gruporegiva>'+RegistroIVA+'</gruporegiva>'+'<IVA>'+
               TextoIVA+'</IVA>'+'</TPV_AltaCliente>');
ConectorWeb.FinCuerpo;
ConectorWeb.FinEstructura;
ConectorWeb.FinMensaje;
Si EncabezadoWeb.Error< >'' entonces
   TextoError=EncabezadoWeb.Error;
```

```
   Limpiar(EncabezadoWeb);
   TablaCliente se inicializa
   TablaCliente se filtra No sea Codigo
   Si encuentra registro TablaCliente entonces
      TablaCliente.ErrorEnvio=Verdadero
      TablaCliente.EnviadoCentral=Falso
      Modificar registro TablaCliente
   Limpiar(EncabezadoWeb)
   Salir(FALSO);
Sino
   TablaCliente se inicializa
   TablaCliente se filtra No sea Codigo
   Si encuentra registro TablaCliente entonces
      TablaCliente.ErrorEnvio=Falso
      TablaCliente.EnviadoCentral=Verdadero
      Modificar registro TablaCliente
   Limpiar(EncabezadoWeb)
   Salir(Verdadero);
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF Conf.FINDFIRST() THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
  END
ELSE
  ERROR('No Existe Configuración TPV');

Cliente.GET(Codigo);
TextoIva := FORMAT(Cliente."Prices Including VAT");

BEGIN
 IF ISCLEAR(SoapHttpConn) THEN
   CREATE(SoapHttpConn);
 SoapHttpConn.Property('EndPointURL', URLWS);
 SoapHttpConn.Connect;
 SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_AltaCliente');
 SoapHttpConn.BeginMessage;
 IF ISCLEAR(SoapSerialize) THEN
   CREATE(SoapSerialize);
 SoapSerialize.Init(SoapHttpConn.InputStream);
 SoapSerialize.StartEnvelope('','STANDARD','');
 SoapSerialize.StartBody('');

 SoapSerialize.WriteXML('<TPV_AltaCliente xmlns="http://tempuri.org/">'+
                  '<Usuario>'+Usuario+'</Usuario>'+
                  '<password>'+Password+'</password>'+
                  '<Numero>'+Codigo +'</Numero>'+
                  '<Nombre>'+Nombre +'</Nombre>'+
                  '<Direccion>'+Direccion +'</Direccion>'+
```

```
                '<Cp>'+Cp +'</Cp>'+
                '<Poblacion>'+Poblacion +'</Poblacion>'+
                '<Provincia>'+Provincia +'</Provincia>'+
                '<Telefono>'+ Telefono +'</Telefono>'+
                '<Telefono2>'+ Telefono2 +'</Telefono2>'+
                '<Email>'+Email +'</Email>'+
                '<NIF>'+CIF +'</NIF>'+
                '<CodFormaPago>'+CodFormaPago +'</CodFormaPago>'+
                '<CodVendedor>'+CodVendedor +'</CodVendedor>'+
                '<CodTerminosPago>'+CodTerminosPago +'</CodTerminosPago>'+
                '<GrupoContCliente>'+GrupoContCliente +'</GrupoContCliente>'+
                '<GrupoContNegocio>'+GrupoContNegocio +'</GrupoContNegocio>'+
                '<GrupoRegIVANeg>'+GrupoRegIVANeg +'</GrupoRegIVANeg>'+
                '<IVA>' + TextoIva + '</IVA>' +
                '</TPV_AltaCliente>');

 SoapSerialize.EndBody;
 SoapSerialize.EndEnvelope;
 SoapHttpConn.EndMessage;
 IF  SoapHttpConn.Error <> '' THEN
    BEGIN
      TextoError:=SoapHttpConn.Error;
      CLEAR(SoapHttpConn);
      Cliente.RESET;
      Cliente.SETRANGE(Cliente."No.",Codigo);
      IF Cliente.FINDFIRST() THEN BEGIN
        Cliente."Error envío":=TRUE;
        Cliente."Enviado a CENTRAL":=FALSE;
        Cliente.MODIFY;
      END;
      CLEAR(SoapHttpConn);
      EXIT(FALSE)
    END
 ELSE
    BEGIN
     Cliente.RESET;
     Cliente.SETRANGE(Cliente."No.",Codigo);
     IF Cliente.FINDFIRST() THEN BEGIN
       Cliente."Error envío":=FALSE;
       Cliente."Enviado a CENTRAL":=TRUE;
       Cliente.MODIFY;
     END;
     CLEAR(SoapHttpConn);
     EXIT(TRUE);
    END;
END;
```

TPV_Sincronizar_Vendedores(Usuario,Password,FechaDesde,FechaHasta)

Esta función se utiliza para actualizar los vendedores que se hayan dado de alta o se hayan modificado en el rango de fechas que se ha indicado en la función. Basicamente, lo que se quiere es que los vendedores tengan un número asociado para que el cálculo de las comisiones sea sencillo para el sistema.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si EncabezadoWeb.Vacio entonces
    Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_Sincronizar_
                  Vendedores');
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXML('<TPV_Sincronizar_Vendedores xmlns="http://
                        tempuri.org/">'+'<usuario>'+Usuario+'</usuario>'+
                        '<password>'+Password+'</password>'+
                        '<fechadesde>'+FechaDesde+'</fechadesde>'+
                        '<fechahasta>'+FechHasta+'</fechahasta>'+
                        ¡</TPV_Sincronizar_Vendedores>');
ConectorWeb.FinCuerpo;
ConectorWeb.FinEstructura;
ConectorWeb.FinMensaje;
Si EncabezadoWeb.Error<>'' entonces
   Limpiar(EncabezadoWeb);
   ERROR('ERROR de comunicaciones al actualizar vendedores, Inténtelo mas tarde')
Si Vacio(NodoXml) entonces
    Crear(NodoXml)
NodoXml.cargar(EncabezadoWeb);
NodoXml.Guardar('Vendedores.xml');
NodoLista=NodoXml.CogerElementos('TablaVendedor.Codigo');
Si NodoLista.Longitud=0 entonces
     Salir(FALSO);
i=0;
Repetir
   NodoLista=NodoXml.CogerElementos('TablaVendedor.Codigo');
   Codigo=NodoLista.elemento(i).texto
   TablaVendedor se inicializa
   TablaVendedor se filtra Vendedor sea Codigo
   Si encuentra registro TablaVendedor entonces
      NodoLista=NodoXml.CogerElemento(TablaVendedor.Nombre);
      TablaVendedor.Nombre=NodoLista.elemento(i).texto;
      NodoLista=NodoXml.CogerElemento(TablaVendedor.Comision);
      Si NodoLista.elemento(i).texto<>'' entonces
         Comision=FormatearImporteDec(NodoLista.elemento(i).texto)
         Evaluar(TablaVendedor.Comision,Comision);
```

```
      NodoLista=NodoXml.CogerElemento(TablaVendedor.FechaModificacion);
      Evaluar(FechaRegistro,CopiaCadena(NodoLista.elemento(i).texto,9,2)+'/'+
      CopiaCadena(NodoLista.elemento(i).texto,6,2)+'/'+CopiaCadena(
      NodoLista.Elemento(i).texto,1,4));
      TablaVendedor.FechaModificacion=FechaRegistro;
      Modificar registro TablaVendedor;
   Sino
      NodoLista=NodoXml.CogerElementos('TablaVendedor.Codigo');
      TablaVendedor.Codigo= NodoLista.elemento(i).texto;
      NodoLista=NodoXml.CogerElemento(TablaVendedor.Nombre);
      TablaVendedor.Nombre=NodoLista.elemento(i).texto;
      NodoLista=NodoXml.CogerElemento(TablaVendedor.Comision);
      Si NodoLista.elemento(i).texto<>'' entonces
         Comision=FormatearImporteDec(NodoLista.elemento(i).texto)
         Evaluar(TablaVendedor.Comision,Comision);
      NodoLista=NodoXml.CogerElemento(TablaVendedor.FechaModificacion);
      Evaluar(FechaRegistro,CopiaCadena(NodoLista.elemento(i).texto,9,2)+'/'+
      CopiaCadena(NodoLista.elemento(i).texto,6,2)+'/'+CopiaCadena(
      NodoLista.Elemento(i).texto,1,4));
      TablaVendedor.FechaModificacion=FechaRegistro;
      Insertar registro TablaVendedor;
   i:=i+1;
Hasta i=NodoLista.Longitud;
Salir(NodoLista.Longitud);
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF Conf.FINDFIRST() THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
   END
ELSE
  ERROR('No Existe Configuración TPV');
BEGIN
 Window.OPEN(
   Text000 +
  '@1@@@@@@@@@@@@@@@@@@@@@@@\');
   Window.UPDATE(1,0);

 IF ISCLEAR(SoapHttpConn) THEN
    CREATE(SoapHttpConn);

 SoapHttpConn.Property('EndPointURL', URLWS);
 SoapHttpConn.Connect;
SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_Sincronizar_Vendedores');
 SoapHttpConn.BeginMessage;
 IF ISCLEAR(SoapSerialize) THEN
    CREATE(SoapSerialize);
 SoapSerialize.Init(SoapHttpConn.InputStream);
 SoapSerialize.StartEnvelope('','STANDARD','');
```

```
  SoapSerialize.StartBody('');

  SoapSerialize.WriteXML('<TPV_Sincronizar_Vendedores xmlns="http://tempuri.org/">'+
                    '<Usuario>'+Usuario+'</Usuario>'+
                    '<password>'+Password+'</password>'+
                    '<FechaDesde>'+FechaDesde +'</FechaDesde>'+
                    '<FechaHasta>'+FechaHasta +'</FechaHasta>'+
                     '</TPV_Sincronizar_Vendedores>');

  SoapSerialize.EndBody;
  SoapSerialize.EndEnvelope;
  SoapHttpConn.EndMessage;
  IF  SoapHttpConn.Error <> '' THEN BEGIN
     CLEAR(SoapHttpConn);
     ERROR(Text0003);
  END;
  IF ISCLEAR(XMLDom) THEN
    CREATE(XMLDom);
  XMLDom.async := FALSE;
  XMLDom.load(SoapHttpConn.OutputStream);
  XMLDom.save(cduILR.DameDirXML() + 'Vendedores.xml');
  DOMNodelist:= XMLDom.getElementsByTagName('Salesperson_x002F_PurchaserCode');
  IF DOMNodelist.length = 0 THEN
    EXIT(0);
  i:=0;


REPEAT
    DOMNodelist:= XMLDom.getElementsByTagName('Salesperson_x002F_PurchaserCode');
    mCod:=DOMNodelist.item(i).text;
    Vendedor.RESET;
    Vendedor.SETRANGE(Vendedor.Code,mCod);
    IF Vendedor.FINDFIRST() THEN BEGIN
      DOMNodelist:=
XMLDom.getElementsByTagName('Salesperson_x002F_PurchaserName');
      Vendedor.Name:=DOMNodelist.item(i).text;
      DOMNodelist:=
XMLDom.getElementsByTagName('Salesperson_x002F_PurchaserCommission');
      IF DOMNodelist.item(i).text <> '' THEN BEGIN
        cvComision:=FormatearImporteDEC(DOMNodelist.item(i).text);
        EVALUATE(Vendedor."Commission %",cvComision);
      END;
      DOMNodelist:=
XMLDom.getElementsByTagName('Salesperson_x002F_PurchaserFechaModificacion');
EVALUATE(FechaRegistro,COPYSTR(DOMNodelist.item(i).text,9,2)+'/'+COPYSTR(DOMN
odelist.item(i).text,6,2)+'/'+
      COPYSTR(DOMNodelist.item(i).text,1,4));
      Vendedor."Fecha de modificación":=FechaRegistro;

      Window.UPDATE(1,ROUND(i / DOMNodelist.length * 10000,1));
      Vendedor.MODIFY;
    END ELSE BEGIN
      Vendedor.INIT;
```

```
      DOMNodelist:=
XMLDom.getElementsByTagName('Salesperson_x002F_PurchaserCode');
      Vendedor.Code:=DOMNodelist.item(i).text;
      DOMNodelist:=
XMLDom.getElementsByTagName('Salesperson_x002F_PurchaserName');
      Vendedor.Name:=DOMNodelist.item(i).text;
      DOMNodelist:=
XMLDom.getElementsByTagName('Salesperson_x002F_PurchaserCommission');
      IF DOMNodelist.item(i).text <> '' THEN BEGIN
        cvComision:=FormatearImporteDEC(DOMNodelist.item(i).text);
        EVALUATE(Vendedor."Commission %",cvComision);
      END;
      DOMNodelist:=
XMLDom.getElementsByTagName('Salesperson_x002F_PurchaserFechaModificacion');
EVALUATE(FechaRegistro,COPYSTR(DOMNodelist.item(i).text,9,2)+'/'+COPYSTR(DOMN
odelist.item(i).text,6,2)+'/'+
      COPYSTR(DOMNodelist.item(i).text,1,4));
      Vendedor."Fecha de modificación":=FechaRegistro;
      Window.UPDATE(1,ROUND(i / DOMNodelist.length * 10000,1));
      Vendedor.INSERT;
    END;
    i:=i+1;
  UNTIL i= DOMNodelist.length;
  Window.CLOSE;
  EXIT(DOMNodelist.length);
END;
```

TPV_Sincronizar_Productos(Usuario,Password,FechaDesde,FechaHasta)

Esta función actualiza los productos en la tienda que se han modificado en el rango de fechas indicado. El motivo de la modificación del producto puede ser: cambio descripción, variación de precio, creación de un nuevo producto o modificación de la familia a la que pertenece. En este caso, se trae únicamente los códigos de los productos y para cada uno invoca a las funciones Sincroniza_Producto y Sincroniza_Tarifa para actualizar los datos del producto y su precio respectivamente.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si EncabezadoWeb.Vacio entonces
    Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/
```

```
                    TPV_Sincronizar_Traer_Productos');
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXML('<TPV_Sincronizar_Traer_Productos xmlns=
                "http://tempuri.org/>'+ '<Usuario>'+Usuario+'</Usuario>'+
                '<password>'+PassWord+'</password>'+'<fechadesde>'+
                FechaDesde+'</fechadesde>'+'<fechahasta>'+FechaHasta+
                '</fechahasta>'+'</TPV_Sincronizar_Traer_Productos>');
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>'' entonces
   Limpiar(EncabezadoWeb);
   Error('ERROR de comunicaciones al actualizar Productos, Inténtelo mas tarde');
Si EstaLimpio(NodoXml) entonces
    Crear(NodoXml)
NodoXml.Cargar(EncabezadoWeb)
NodoXml.Guardar('Productos.xml');
NodoLista=NodoXml.CogerElementos(TablaProducto.No)
Si NodoLista.Longitud=0 entonces
    Salir(FALSO);
Sino
    Cuantos=NodoLista.Longitud;
i=0
Repetir
   NodoLista=NodoXml.CogerElemento(TablaProducto.No);
   Cod=NodoLista.elemento(i).texto
   TPV_Sincronizar_Producto(Cod);
   TPV_Sincronizar_Tarifa(Cod)
   i=i+1;
Hasta i=Cuantos
Exit(Verdadero)
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF Conf.FINDFIRST() THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
   END
ELSE
  ERROR('No Existe Configuración TPV');
BEGIN
 Ven.UPDATE(3,0);
 IF ISCLEAR(SoapHttpConn) THEN
    CREATE(SoapHttpConn);
 SoapHttpConn.Property('EndPointURL', URLWS);
```

```
  SoapHttpConn.Connect;

  SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_Sincronizar_Traer_Productos');
  SoapHttpConn.BeginMessage;
  IF ISCLEAR(SoapSerialize) THEN
    CREATE(SoapSerialize);
  SoapSerialize.Init(SoapHttpConn.InputStream);
  SoapSerialize.StartEnvelope('','STANDARD','');
  SoapSerialize.StartBody('');

  SoapSerialize.WriteXML('<TPV_Sincronizar_Traer_Productos
xmlns="http://tempuri.org/">'+
                    '<Usuario>'+Usuario+'</Usuario>'+
                    '<password>'+Password+'</password>'+
                    '<FechaDesde>'+FechaDesde +'</FechaDesde>'+
                    '<FechaHasta>'+FechaHasta +'</FechaHasta>'+
                     '</TPV_Sincronizar_Traer_Productos>');

  SoapSerialize.EndBody;
  SoapSerialize.EndEnvelope;
  SoapHttpConn.EndMessage;
  IF  SoapHttpConn.Error <> '' THEN BEGIN
      CLEAR(SoapHttpConn);
      ERROR(Text0004);
  END;

  IF ISCLEAR(XMLDom) THEN
    CREATE(XMLDom);
  XMLDom.async := FALSE;
  XMLDom.load(SoapHttpConn.OutputStream);
  XMLDom.save(cduILR.DameDirXML() + 'Productos.xml');

  DOMNodelist:= XMLDom.getElementsByTagName('ItemNo');
  IF DOMNodelist.length = 0 THEN
    EXIT(0)
  ELSE
    Cuantos := DOMNodelist.length;
  Ven.UPDATE(2,FORMAT(Cuantos));
  i:=0;
  REPEAT
    DOMNodelist:= XMLDom.getElementsByTagName('ItemNo');
    mCod:=DOMNodelist.item(i).text;
    TPV_Sincronizar_Producto(mCod);
    TPV_Sincronizar_Tarifa(mCod);
    i:=i+1;
    Ven.UPDATE(3,ROUND(i / Cuantos * 10000,1));
  UNTIL i= Cuantos;
  EXIT(Cuantos);
END;
```

TPV_Sincronizar_Tiendas(Usuario,Password)

Esta función sirve para actualizar el listado de tiendas, debido a que se han podido abrir nuevas tiendas, crear nuevos almacenes destino para las transferencias, cambiar el

tipo de almacén, etc. En todo momento, se debe tener actualizado el listado para que luego no haya problemas a la hora de realizar transferencias.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si EncabezadoWeb.Vacio entonces
    Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_Sicronizar_Tiendas);
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXML('<TPV_Sincronizar_Tiendas xmlns= "http://tempuri.org/">
                +'<Usuario>'+Usuario+'</Usuario>'+'<password>'+Password+
                '</password>'+'</TPV_Sincronizar_Tiendas>');
ConectorWeb.FinCuerpo;
ConectorWeb.FinEstructura;
EncabezadoWeb.FinMensaje;
Si EncabezadoWeb.Error<>'' entonces
   Limpiar(EncabezadoWeb);
   ERROR('ERROR de comunicaciones al actualizar Tiendas, Inténtelo mas tarde')
Si EstaVacio(NodoXml) entonces
    Crear(NodoXml)
NodoXml.Cargar(EncabezadoWeb)
NodoXml.Guardar('Tiendas.xml')
NodoLista=NodoXml.CogerElementos('TablaAlmacen.Codigo');
Si NodoLista.Longitud=0 entonces
   Salir(FALSO)
TablaTienda se inicializa
Se borran todos los registros TablaTienda
i=0
Repetir
    NodoLista=NodoXml.CogerElemento(TablaAlmacen.Codigo)
    Codigo=NodoLista.elemento(i).texto
    TablaTienda se inicializa
    TablaTienda se filtra Codigo sea Codigo
    Si encuentra registro TablaTienda entonces
       NodoLista=NodoXml.CogerElemento(TablaAlmancen.Nombre)
       TablaTienda.Nombre=NodoLista.elemento(i).texto
       NodoLista=NodoXml.CogerElemento(TablaAlmacen.TipoAlmacen)
       TipoTienda=NodoLista.elemento(i).texto
```

```
        Caso TipoTienda
           Tienda: TablaTienda.TipoAlmacen=Tienda
           Central: TablaTienda.TipoAlmacen=Central
           Franquicia: TablaTienda.TipoAlmacen=Franquicia
           Ferias: TablaTienda.TipoAlmacen=Ferias
           Otros: TablaTienda.TipoAlmacen=Otros
        NodoLista=NodoXml.CogerElemento(TablaAlmacen.CodigoEquivalencia)
        TablaTienda.CodigoEquivalencia=NodoLista.elemento(i).texto
        Modificar registro TablaTienda
     Sino
        TablaTienda se inicializa
        NodoLista=NodoXml.CogerElemento(TablaAlmacen.Codigo)
        TablaTienda.Codigo=NodoLista.elemento(i).texto
        NodoLista=NodoXml.CogerElemento(TablaAlmacen.Nombre)
        TablaTienda.Nombre=NodoLista.elemento(i).texto
        NodoLista=NodoXml.CogerElemento(TablaAlmacen.TipoAlmacen)
        TipoTienda=NodoLista.elemento(i).texto
        Caso TipoTienda
           Tienda: TablaTienda.TipoAlmacen=Tienda
           Central: TablaTienda.TipoAlmacen=Central
           Franquicia: TablaTienda.TipoAlmacen=Franquicia
           Ferias: TablaTienda.TipoAlmacen=Ferias
           Otros: TablaTienda.TipoAlmacen=Otros
        NodoLista=NodoXml.CogerElemento(TablaAlmacen.CodigoEquivalencia)
        TablaTienda.CodigoEquivalencia=NodoLista.elemento(i).texto
        Insertar registro TablaTienda
Hasta i=NodoLista.Longitud
Salir(Verdadero)
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF Conf.FINDFIRST() THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
  END
ELSE
  ERROR('No Existe Configuración TPV');
BEGIN
 Window.OPEN(
   Text000 +
  '@1@@@@@@@@@@@@@@@@@@@@@@@\');
   Window.UPDATE(1,0);
 IF ISCLEAR(SoapHttpConn) THEN
    CREATE(SoapHttpConn);

 SoapHttpConn.Property('EndPointURL', URLWS);
 SoapHttpConn.Connect;

 SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_Sincronizar_Tiendas');
 SoapHttpConn.BeginMessage;
```

```
  IF ISCLEAR(SoapSerialize) THEN
    CREATE(SoapSerialize);
  SoapSerialize.Init(SoapHttpConn.InputStream);
  SoapSerialize.StartEnvelope('','STANDARD','');
  SoapSerialize.StartBody('');

  SoapSerialize.WriteXML('<TPV_Sincronizar_Tiendas xmlns="http://tempuri.org/">'+
                         '<Usuario>'+Usuario+'</Usuario>'+
                         '<password>'+Password+'</password>'+
                          '</TPV_Sincronizar_Tiendas>');

  SoapSerialize.EndBody;
  SoapSerialize.EndEnvelope;
  SoapHttpConn.EndMessage;

  IF  SoapHttpConn.Error <> '' THEN BEGIN
     CLEAR(SoapHttpConn);
     ERROR(Text0006);
  END;

  IF ISCLEAR(XMLDom) THEN
    CREATE(XMLDom);
  XMLDom.async := FALSE;
  XMLDom.load(SoapHttpConn.OutputStream);
  XMLDom.save(cduILR.DameDirXML() + 'Tiendas.xml'); // Esto es para que veas el XML de
Respuesta
  DOMNodelist:= XMLDom.getElementsByTagName('LocationCode');
  IF DOMNodelist.length = 0 THEN
    EXIT(0);
  Tienda.RESET;
  Tienda.DELETEALL();
  i:=0;
  REPEAT
    DOMNodelist:= XMLDom.getElementsByTagName('LocationCode');
    mCod:=DOMNodelist.item(i).text;
    Tienda.RESET;
    Tienda.SETRANGE(Tienda.Code,mCod);
    IF Tienda.FINDFIRST() THEN BEGIN
      DOMNodelist:= XMLDom.getElementsByTagName('LocationName');
      Tienda.Name:=DOMNodelist.item(i).text;
      DOMNodelist:= XMLDom.getElementsByTagName('LocationTipoAlmacen');
      TipoTienda := DOMNodelist.item(i).text;
      CASE TipoTienda OF
        'Tienda'     : Tienda."Tipo Almacen" := Tienda."Tipo Almacen"::Tienda;
        'Central'    : Tienda."Tipo Almacen" := Tienda."Tipo Almacen"::Central;
        'Franquicia' : Tienda."Tipo Almacen" := Tienda."Tipo Almacen"::Franquicia;
        'Ferias'     : Tienda."Tipo Almacen" := Tienda."Tipo Almacen"::Ferias;
        'Otros'      : Tienda."Tipo Almacen" := Tienda."Tipo Almacen"::Otros;
      END;
      DOMNodelist:= XMLDom.getElementsByTagName('LocationCódEquivalenciaAntiguo');
      Tienda."Cód. Equivalencia Antiguo" := DOMNodelist.item(i).text;
      Window.UPDATE(1,ROUND(i / DOMNodelist.length * 10000,1));
      Tienda.MODIFY;
    END ELSE BEGIN
```

```
      Tienda.INIT;
      DOMNodelist:= XMLDom.getElementsByTagName('LocationCode');
      Tienda.Code:=DOMNodelist.item(i).text;
      DOMNodelist:= XMLDom.getElementsByTagName('LocationName');
      Tienda.Name:=DOMNodelist.item(i).text;
      DOMNodelist:= XMLDom.getElementsByTagName('LocationTipoAlmacen');
      TipoTienda := DOMNodelist.item(i).text;
      CASE TipoTienda OF
        'Tienda'      : Tienda."Tipo Almacen" := Tienda."Tipo Almacen"::Tienda;
        'Central'     : Tienda."Tipo Almacen" := Tienda."Tipo Almacen"::Central;
        'Franquicia'  : Tienda."Tipo Almacen" := Tienda."Tipo Almacen"::Franquicia;
        'Ferias'      : Tienda."Tipo Almacen" := Tienda."Tipo Almacen"::Ferias;
        'Otros'       : Tienda."Tipo Almacen" := Tienda."Tipo Almacen"::Otros;
      END;
      DOMNodelist:= XMLDom.getElementsByTagName('LocationCódEquivalenciaAntiguo');
      Tienda."Cód. Equivalencia Antiguo" := DOMNodelist.item(i).text;
      Window.UPDATE(1,ROUND(i / DOMNodelist.length * 10000,1));
      Tienda.INSERT;
    END;
    i:=i+1;
  UNTIL i= DOMNodelist.length;
  Window.CLOSE;
  EXIT(DOMNodelist.length);
END;
```

TPV_Factura_Generar(Usuario,Password,Ticket,Envio,Cliente,EnvioNombre, EnvioDireccion,EnvioCP,EnvioPoblacion,EnvioProvincia,EnvioAtencion, CodTienda,CodFormaPago,CodVendedor);

Esta función sirve para mandar la factura con todos los datos (cabecera, líneas y formas de pago) a CENTRAL. Entonces, esta función se invoca una vez que el sistema ha registrado, de forma local, la factura y así poder enviarla a CENTRAL y que los datos queden centralizados.

PseudoCódigo

```
TablaTicket se inicializa
TablaTicket se filtra TipoDocumento sea Factura
TablaTicket se filtra NoDocumento sea Ticket
Si no encuentra registros TablaTicket entonces
   ERROR('No existe información del Ticket');
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
FechaTicket=Formatear(TablaTicket.FechaDocumento);
CodTpv=TablaTicket.CodTpv
```

```
HoraTicket=Formatear(TablaTicket.HoraDocumento)
TotalaPagar=TablaTicket.TotalaPagar
ImporteEntegado=Formatear(TablaTicket.ImporteEntregado)
ImporteVenta=Formatear(TablaTicket.TotalVentaProvisional)
Si EstaVacio(NodoXml) entonces
   Crear(NodoXml)
CadenaInicial='<NewDataSet>';
CadenaLineaInicial='<Lineas>'
CadenaFinal='</NeweDataSet>'
CadenaLineaFinal='</Lineas>'
CadenaTipoIni='<Tipo>'
CadenaTipoFin='</Tipo>'
CadenaQtyIni='<Quantity>'
CadenaQtyFin='</Quantity>'
CadenaPrecioIni='<UnitPrice>'
CadenaPrecioFin='</UnitPrice>'
CadenaDescIni='<LineDiscount>'
CadenaDescFin='</LineDiscount>'
CadenaDescripIni='<Description>'
CadenaDescripFin='</Description>'
CadenaLineaIni='<Linea>'
CadenaLineaFin='</Linea>'
CadenaFormaPagoIni='<FormPago>'
CadenaFormaPagoFin='</FormPago>'
CadenaImporteIni='<Importe>'
CadenaImporteFin='</Importe>'
CadenaCambioIni='<Cambio>'
CadenaCambioFin='</Cambio>'
CadenaValeIni='<Vale>'
CadenaValeFin='</Vale>'
CadenaTipoPagoIni='<TipoPago>'
CadenaTipoPagoFin='</TipoPago>'
CadenaCobradoIni='<Cobrado>'
CadenaCobradoFin='</Cobrado>'
CadenaValeResIni='<ValeRes>'
CadenaValeResFin='</ValeRes>'
CabeceraVenta se inicializa
CabeceraVenta se filtra NoPreAsignado sea Ticket
Si encuentra registro CabeceraVenta entonces
    NumeroDoc=CabeceraVenta.No
Sino
    ERROR('ERROR, no encuentro la cabecera venta en el histórico);
TextoIVA=Formatear(CabeceraVenta.PrecioIncluyeIVA);
LineasVenta se inicializa
LineasVenta se filtra NoDocumento sea NumeroDoc
Si encuentra registros LineasVenta entonces
   i=1
   Repetir
      Cadena(i)=Cadena(i)+CadenaLineaInicial
      Cadena(i)=Cadena(i)+CadenaTipoIni+Formatear(LineasVenta.Tipo)+
                CadenaTipFin
      Cadena(i)=Cadena(i)+CadenaNumeroIni+LineasVenta.No+CadenaNumeroFin
      Cadena(i)=Cadena(i)+CadenaQtyIni+Formateo(LineasVenta.Cantidad)+
                CadenaQtyFin
```

```
      Cadena(i)=Cadena(i)+CadenaPrecioIni+Formateo(LineasVenta.Precio)+
                  CadenaPrecioFin
      Cadena(i)=Cadena(i)+CadenaDescIni+Formateo(LineasVenta.Descuento)+
                  CadenaDescFin
      tempDesc=LineasVenta.Descripcion
     ConvertDesc=''
     fin=FALSO
     Mientras NO fin hacer
        iPos=PosicionCadena(tempDesc,'&')
       Si iPos=0 entonces
          ConvertDesc=ConvertDesc+tempDesc
          Fin=Verdadero
       Sino
          ConvertDesc=ConvertDesc+CopiaCadena(tempDesc,1,iPos-1)+'&amp;'
          tempDesc=CopiaCadena(tempDesc,iPos+1)
      Cadena(i)=Cadena(i)+CadenaDescripIni+ConvertDesc+CadenaDescripFin
      Cadena(i)=Cadena(i)+CadenaLineaFinal
   Hasta siguiente registro LineasVenta sea vacio
Multiforma se inicializa
Multiforma se filtra TipoDocumento sea Factura
Multiforma se filtra NoDocumento sea Ticket
z=1
Si encuentra registros Multiforma entonces
  Repetir
     CadenaMulti(z)=CadenaMulti(z)+CadenaLineaInicial
     CadenaMulti(z)=CadenaMulti(z)+CadenaLineaIni+Formateo(Multiforma.NoLinea)
                    +CadenaLineaFin
     CadenaMulti(z)=CadenaMulti(z)+CadenaFormPagoIni+Multiforma.CodFormaPago
                     +CadenaFormPagoFin
     CadenaMulti(z)=CadenaMulti(z)+CadenaImporteIni+Format(Multiforma.Importe)+
                     CadenaImporteFin
     CadenaMulti(z)=CadenaMulti(z)+CadenaCambioIni+Format(Multiforma.Cambio)+
                     CadenaCambioFin
     CadenaMulti(z)=CadenaMulti(z)+CadenaValeIni+Multiforma.NoValeAbono+
                     CadenaValeFin
     CadenaMulti(z)=CadenaMulti(z)+CadenaTipoPagoIni+Multiforma.TipoPago+
                     CadenaTipoPagoFin
     CadenaMulti(z)=CadenaMulti(z)+CadenaCobradoIni+Multiforma.Cobrado+
                     CadenaCobradoFin
     CadenaMulti(z)=CadenaMulti(z)+CadenaValeResIni+Multiforma.Abono+
                     CadenaValeResFin
     CadenaMulti(z)=CadenaMulti(z)+'<TipoVale>'+Multiforma.TipoVale+
                     '</TipoVale>'
     CadenaMulti(z)=CadenaMulti(z)+CadenaLineaFinal
     z=z+1
  Hasta siguiente registro Multiforma sea vacío
Si EstaVacio(EncabezadoWeb) entonces
   Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_Factura_Generar');
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
```

```
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura(‘’,’STANDARD’,’’)
ConectorWeb.InicioCuerpo(‘’);
ConectorWeb.EscribirXML(‘<TPV_Factura_Generar xmlns=”http://tempuri.org/”>’+
                ‘<Usuario>’+Usuario+’</Usuario>’+’<Password>’+Password+
                ‘</password>’+’<NumeroTicket>’+Ticket+’</NumeroTicket>’+
                ‘<Cliente>’+Cliente+’</Cliente>’+’<CodTienda>’+CodTienda+
                ‘</CodTienda>’+’<CodDirEnvio>’+CodigoDireccionEnvio+
                ‘</CodDirEnvio>’+’<EnvioNombre>’+EnvioNombre+’</EnvioNombre>
                +’<EnvioDireccion>’+EnvioDireccion+’</EnvioDireccion>’+
                +’<EnvioCP>’+EnvioCP+’</EnvioCP>’+’<EnvioPoblacion>’+
                EnvioPoblacion+’</EnvioPoblacion>’+’<EnvioProvincia>’+
                EnvioProvincia+’</EnvioProvincia>’+’<EnvioAtencion>’+
                EnvioAtencion+’</EnvioAtencion>’+’<CodFormaPago>’+
                CodFormaPago+’</CodFormaPago>’+’<CodVendedor>’+CodVendedor
                +’</CodVendedor>’+’<FechaTicket>’+FechaTicket+’</FechaTicket>’+
                ‘<CodTpv>’+CodTPV+’</CodTpv>’+’<HoraTicket>’+HoraTicket+
                ‘</HoraTicket>’+’<TotalAPagar>’+TotalAPagar+’</TotalAPagar>’+
                ‘<ImporteVenta>’+Recibido+’</ImporteVenta>’+’<ImporteEntregado>’
                +CabeceraVenta.ImporteEntregado+’</ImporteEntregado>’+
                ’<NumeroReserva>’+CabeceraVenta.NoReserva+’</NumeroReserva>’+
                ‘<IVAIncluido>’+TextoIVA+’</IVAIncluido>’+’<Tarifa>’+
                CabeceraVenta.GrupoPrecioIVA+’</Tarifa>’+’<DatosLineasStr>’);
ConectorWeb.EscribirCadena(CadenaInicial);
x=1
Repetir
    ConectorWeb.EscribirCadena(Cadena(x))
    x=x+1
Hasta x>i;
ConectorWeb.EscribirCadena(CadenaFinal)
ConectorWeb.EscribirXML(‘</DatosLineasStr><DatosMultiStr>’)
ConectorWeb.EscribirCadena(CadenaInicial)
x=1
Repetir
   ConectorWeb.EscribirCadena(CadenaMulti(x))
   x=x+1
Hasta x>z
ConectorWeb.EscribirCadena(CadenaFinal)
ConectorWeb.EscribirXML(‘</DatosMultiStr></TPV_Factura_Generar>’)
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>’’ entonces
   TxtError=EncabezadoWeb.Error
   Salir(Falso)
Si EstaVacio(NodoXml) entonces
    Crear(NodoXml)
NodoXml.cargar(EncabezadoWeb)
NodoXml.guardar(‘Ticket.xml’)
Si ComprobarError(NodoXml,TxtError) entonces
    Salir(Falso)
Salir(Verdadero)
```

Código Original

```
RecTicket.RESET;
RecTicket.SETRANGE(RecTicket."Tipo documento",RecTicket."Tipo documento"::Factura);
RecTicket.SETRANGE(RecTicket."Nº documento",Ticket);
IF NOT RecTicket.FINDFIRST() THEN
  ERROR('No existe Información de Ticket');

Conf.RESET;
Conf.SETRANGE("Cód. tienda", RecTicket."Cód. tienda");
Conf.SETRANGE("Cód. TPV", RecTicket."Cód. TPV");
IF Conf.FINDFIRST() THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
   END
ELSE
  ERROR('No Existe Configuración TPV');

FechaTicket        :=FORMAT(RecTicket."Fecha documento");
CodTPV             :=RecTicket."Cód. TPV";
HoraTicket         :=FORMAT(RecTicket."Hora documento");
TotalApagar        :=FORMAT(RecTicket."Total a pagar");
ImporteEntregado   :=FORMAT(RecTicket."Importe entregado");
ImporteVenta       :=FORMAT(RecTicket."Total venta provisional");

 BEGIN
    IF ISCLEAR(XMLDom) THEN
      CREATE (XMLDom);

    CadenaInicial:='<NewDataSet>';
    CadenaLineaInicial:='<Lineas>';
    CadenaFinal:='</NewDataSet>';
    CadenaLineaFinal:='</Lineas>';

    CadenaTipoIni:='<Tipo>';
    CadenaTipoFin:='</Tipo>';


    CadenaNumeroIni:='<No.>';
    CadenaNumeroFin:='</No.>';

    CadenaQtyIni:='<Quantity>';
    CadenaQtyFin:='</Quantity>';

    CadenaUnitPriceIni:='<UnitPrice>';
    CadenaUnitPriceFin:='</UnitPrice>';

    CadenaLineDisIni:='<LineDiscount>';
    CadenaLineDisFin:='</LineDiscount>';
    CadenaDescriptionIni := '<Description>';
    CadenaDescriptionFin := '</Description>';
```

```
    CadenaLineaini    :='<Linea>';
    CadenaLineaFIni   :='</Linea>';
    CadenaFormPagoIni :='<FormPago>';
    CadenaFormPagoFin :='</FormPago>';
    CadenaImporteIni  :='<Importe>';
    CadenaImporteFin  :='</Importe>';
    CadenaCambioIni   :='<Cambio>';
    CadenaCambioFin   :='</Cambio>';
    CadenaValeIni     :='<Vale>';
    CadenaValeFin     :='</Vale>';
    CadenaTipoPagoIni :='<TipoPago>';
    CadenaTipoPagoFin :='</TipoPago>';
    CadenaCobradoIni  :='<Cobrado>';
    CadenaCobradoFin  :='</Cobrado>';
    CadenaValeResIni  :='<ValeRes>';
    CadenaValeResFin  :='</ValeRes>';



    PsalesHeader.RESET;
    PsalesHeader.SETCURRENTKEY("Pre-Assigned No.");
    PsalesHeader.SETRANGE(PsalesHeader."Pre-Assigned No.",Ticket);
    IF PsalesHeader.FINDFIRST() THEN
      NumeroDOC:=PsalesHeader."No."
    ELSE
      ERROR('ERROR no encuentro la cabecera de Venta en el histórico');
    TextoIva := FORMAT(PsalesHeader."Prices Including VAT");

    PsalesLine.RESET;
    PsalesLine.SETRANGE(PsalesLine."Document No.",NumeroDOC);
    IF PsalesLine.FINDSET() THEN
    BEGIN
      i:=1;
     REPEAT
        Cadena[i] := Cadena[i] + CadenaLineaInicial;
        Cadena[i] := Cadena[i] + CadenaTipoIni+FORMAT(PsalesLine.Type)+ CadenaTipoFin;
        Cadena[i] := Cadena[i] + CadenaNumeroIni+PsalesLine."No."+ CadenaNumeroFin;
        Cadena[i] := Cadena[i] +
CadenaQtyIni+FORMAT(PsalesLine.Quantity)+CadenaQtyFin;
        Cadena[i] := Cadena[i] + CadenaUnitPriceIni+FORMAT(PsalesLine."Unit
Price")+CadenaUnitPriceFin;
        Cadena[i] := Cadena[i] + CadenaLineDisIni+FORMAT(PsalesLine."Line Discount
%")+CadenaLineDisFin;

        tempDesc:=PsalesLine.Description;
        ConvertDesc:='';
        fin:=FALSE;
        WHILE NOT fin DO
        BEGIN
          iPos:=STRPOS(tempDesc,'&');
          IF  iPos = 0 THEN
          BEGIN
            ConvertDesc+=tempDesc;
            fin:=TRUE;
```

```
          END
          ELSE
          BEGIN
            ConvertDesc += COPYSTR(tempDesc,1,iPos-1) + '&amp;';
            tempDesc:= COPYSTR(tempDesc,iPos+1);
          END;
        END;

        Cadena[i] := Cadena[i] + CadenaDescriptionIni + ConvertDesc + CadenaDescriptionFin;
        Cadena[i] := Cadena[i] + CadenaLineaFinal;
        i:=i+1;
      UNTIL PsalesLine.NEXT=0;
    END;

    Multiforma.RESET;
    Multiforma.SETRANGE(Multiforma."Tipo                    Documento",Multiforma."Tipo
Documento"::Factura);
    Multiforma.SETRANGE(Multiforma."Nº Documento",Ticket);
    z:=1;

    IF Multiforma.FINDSET() THEN REPEAT
       CadenaMulti[z] := CadenaMulti[z] + CadenaLineaInicial;
       CadenaMulti[z] := CadenaMulti[z] + CadenaLineaini+FORMAT(Multiforma."Nº Linea")
+CadenaLineaFIni;
       CadenaMulti[z] := CadenaMulti[z] + CadenaFormPagoIni+Multiforma."Cod. Forma
Pago" + CadenaFormPagoFin;
       CadenaMulti[z] := CadenaMulti[z] + CadenaImporteIni+FORMAT(Multiforma.Importe)
+CadenaImporteFin;
       CadenaMulti[z] := CadenaMulti[z] + CadenaCambioIni+FORMAT(Multiforma.Cambio)
+CadenaCambioFin;
       CadenaMulti[z] := CadenaMulti[z] + CadenaValeIni+Multiforma."Nº vale de
abono"+CadenaValeFin;
       CadenaMulti[z] := CadenaMulti[z] + CadenaTipoPagoIni+FORMAT(Multiforma."Tipo
pago") +CadenaTipoPagoFin;
       CadenaMulti[z] := CadenaMulti[z] + CadenaCobradoIni+FORMAT(Multiforma.Cobrado)
+CadenaCobradoFin;
       CadenaMulti[z] := CadenaMulti[z] + CadenaValeResIni+Multiforma."Nº bono / Reserva /
Abono"+CadenaValeResFin;
       CadenaMulti[z] := CadenaMulti[z] + '<TipoVale>' + FORMAT(Multiforma."Tipo de
vale") + '</TipoVale>';
       CadenaMulti[z] := CadenaMulti[z] + CadenaLineaFinal;
      z:=z+1;
    UNTIL Multiforma.NEXT=0;

    IF ISCLEAR(SoapHttpConn) THEN
      CREATE(SoapHttpConn);

    SoapHttpConn.Property('EndPointURL', URLWS);
    SoapHttpConn.Connect;

    SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_Factura_Generar');
    SoapHttpConn.BeginMessage;
    IF ISCLEAR(SoapSerialize) THEN
      CREATE(SoapSerialize);
```

```
    SoapSerialize.Init(SoapHttpConn.InputStream);
    SoapSerialize.StartEnvelope('','STANDARD','');
    SoapSerialize.StartBody('');
    SoapSerialize.WriteXML('<TPV_Factura_Generar xmlns="http://tempuri.org/">'+
                    '<Usuario>'+Usuario+'</Usuario>'+
                    '<password>'+Password+'</password>'+
                    '<NumeroTicket>'+Ticket+'</NumeroTicket>'+
                    '<Cliente>'+Cliente+'</Cliente>'+
                    '<CodTienda>'+CodTienda+'</CodTienda>'+
                    '<CodDirEnvio>'+ CodigoDireccionEnvio +'</CodDirEnvio>'+
                    '<EnvioNombre>'+EnvioNombre+'</EnvioNombre>'+
                    '<EnvioDireccion>'+EnvioDireccion+'</EnvioDireccion>'+
                    '<EnvioCP>'+EnvioCP+'</EnvioCP>'+
                    '<EnvioPoblacion>'+EnvioPoblacion+'</EnvioPoblacion>'+
                    '<EnvioProvincia>'+EnvioProvincia+'</EnvioProvincia>'+
                    '<EnvioAtencion>'+EnvioAtencion+'</EnvioAtencion>'+
                    '<CodFormaPago>'+CodFormaPago+'</CodFormaPago>'+
                    '<CodVendedor>'+CodVendedor+'</CodVendedor>'+
                    '<FechaTicket>'+FechaTicket+'</FechaTicket>'+
                    '<CodTPV>'+ CodTPV +'</CodTPV>'+
                    '<HoraTicket>'+HoraTicket+'</HoraTicket>'+
                    '<TotalApagar>'+TotalApagar+'</TotalApagar>'+
                    '<ImporteVenta>'+Recibido+'</ImporteVenta>'+
                    '<ImporteEntregado>'+FORMAT(PsalesHeader."Importe
Entregado")+'</ImporteEntregado>'+
                    '<NumeroReserva>'+PsalesHeader."Nº Reserva"+'</NumeroReserva>'+
                    '<IvaIncluido>' + TextoIva + '</IvaIncluido>' +
                    '<Tarifa>' + PsalesHeader."Customer Price Group" + '</Tarifa>' +
                    '<DatosLineasStr>');

    SoapSerialize.WriteString(CadenaInicial);
    x:=1;
    REPEAT
      SoapSerialize.WriteString(Cadena[x]);
      x:=x+1;
    UNTIL x > i;
    SoapSerialize.WriteString(CadenaFinal);
    SoapSerialize.WriteXML('</DatosLineasStr><DatosMultiStr>');
    SoapSerialize.WriteString(CadenaInicial);
    x:=1;
    REPEAT
      SoapSerialize.WriteString(CadenaMulti[x]);
      x:=x+1;
    UNTIL x > z;
    SoapSerialize.WriteString(CadenaFinal);
    SoapSerialize.WriteXML('</DatosMultiStr></TPV_Factura_Generar>');
    SoapSerialize.EndBody;
    SoapSerialize.EndEnvelope;
    SoapHttpConn.EndMessage;

    IF SoapHttpConn.Error <> '' THEN BEGIN
       TxtError:=SoapHttpConn.Error;
       EXIT(FALSE);
    END;
```

```
    IF ISCLEAR(XMLDom) THEN
      CREATE(XMLDom);

    XMLDom.async := FALSE;
    XMLDom.load(SoapHttpConn.OutputStream);
     XMLDom.save(cduILR.DameDirXML() + 'Ticket.xml');

    IF ComprobarERROR(XMLDom,TxtError) THEN BEGIN
      EXIT(FALSE);
    END;
    EXIT(TRUE);
END;
```

TPV_Abono_Generar(Usuario,Password,Ticket,Envio,Cliente,EnvioNombre, EnvioDireccion,EnvioCP,EnvioPoblacion,EnvioProvincia,EnvioAtencion, CodTienda,CodFormaPago,CodVendedor,NumTicketAbonado);

Esta función sirve para mandar el abono con todos los datos (cabecera, líneas y formas de pago) a CENTRAL. Entonces, esta función se invoca una vez que el sistema ha registrado, de forma local, el abono y así poder enviarlo a CENTRAL y que los datos queden centralizados.

PseudoCódigo

```
TablaTicket se inicializa
TablaTicket se filtra TipoDocumento sea Devolucion
TablaTicket se filtra NoDocumento sea Ticket
Si no encuentra registros TablaTicket entonces
   ERROR('No existe información del Ticket');
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
FechaTicket=Formatear(TablaTicket.FechaDocumento);
CodTpv=TablaTicket.CodTpv
HoraTicket=Formatear(TablaTicket.HoraDocumento)
TotalaPagar=TablaTicket.TotalaPagar
ImporteEntregado=Formatear(TablaTicket.ImporteEntregado)
ImporteDevolucion=Formatear(TablaTicket.TotalVentaProvisional)
Si EstaVacio(NodoXml) entonces
   Crear(NodoXml)
CadenaInicial='<NewDataSet>';
CadenaLineaInicial='<Lineas>'
CadenaFinal='</NeweDataSet>'
CadenaLineaFinal='</Lineas>'
CadenaTipoIni='<Tipo>'
```

```
CadenaTipoFin=’</Tipo>’
CadenaNumeroIni=’<No>’
CadenaNumeroFin=’</No>’
CadenaQtyIni=’<Quantity>’
CadenaQtyFin=’</Quantity>’
CadenaPrecioIni=’<UnitPrice>’
CadenaPrecioFin=’</UnitPrice>’
CadenaDescIni=’<LineDiscount>’
CadenaDescFin=’</LineDiscount>’
CadenaDescripIni=’<Description>’
CadenaDescripFin=’</Description>’
CadenaLineaIni=’<Linea>’
CadenaLineaFin=’</Linea>’
CadenaFormaPagoIni=’<FormPago>’
CadenaFormaPagoFin=’</FormPago>’
CadenaImporteIni=’<Importe>’
CadenaImporteFin=’</Importe>’
CadenaCambioIni=’<Cambio>’
CadenaCambioFin=’</Cambio>’
CadenaValeIni=’<Vale>’
CadenaValeFin=’</Vale>’
CadenaTipoPagoIni=’<TipoPago>’
CadenaTipoPagoFin=’</TipoPago>’
CadenaCobradoIni=’<Cobrado>’
CadenaCobradoFin=’</Cobrado>’
CadenaValeResIni=’<ValeRes>’
CadenaValeResFin=’</ValeRes>’
CabeceraVenta se inicializa
CabeceraVenta se filtra NoPreAsignado sea Ticket
Si encuentra registro CabeceraVenta entonces
    NumeroDoc=CabeceraVenta.No
Sino
    ERROR(‘ERROR, no encuentro la cabecera abono en el histórico);
TextoIVA=Formatear(CabeceraVenta.PrecioIncluyeIVA);
LineasVenta se inicializa
LineasVenta se filtra NoDocumento sea NumeroDoc
Si encuentra registros LineasVenta entonces
   i=1
   Repetir
      Cadena(i)=Cadena(i)+CadenaLineaInicial
      Cadena(i)=Cadena(i)+CadenaTipoIni+Formatear(LineasVenta.Tipo)+
                CadenaTipFin
      Cadena(i)=Cadena(i)+CadenaNumeroIni+LineasVenta.No+CadenaNumeroFin
      Cadena(i)=Cadena(i)+CadenaQtyIni+Formateo(LineasVenta.Cantidad)+
                CadenaQtyFin
      Cadena(i)=Cadena(i)+CadenaPrecioIni+Formateo(LineasVenta.Precio)+
                CadenaPrecioFin
      Cadena(i)=Cadena(i)+CadenaDescIni+Formateo(LineasVenta.Descuento)+
                CadenaDescFin
      tempDesc=LineasVenta.Descripcion
     ConvertDesc=’’
     fin=FALSO
     Mientras NO fin hacer
        iPos=PosicionCadena(tempDesc,’&’)
```

```
        Si iPos=0 entonces
           ConvertDesc=ConvertDesc+tempDesc
           Fin=Verdadero
        Sino
           ConvertDesc=ConvertDesc+CopiaCadena(tempDesc,1,iPos-1)+'&amp;amp;'
           tempDesc=CopiaCadena(tempDesc,iPos+1)
     Cadena(i)=Cadena(i)+CadenaDescripIni+ConvertDesc+CadenaDescripFin
     Cadena(i)=Cadena(i)+CadenaLineaFinal
  Hasta siguiente registro LineasVenta sea vacio
Multiforma se inicializa
Multiforma se filtra TipoDocumento sea Devolucion
Multiforma se filtra NoDocumento sea Ticket
z=1
Si encuentra registros Multiforma entonces
 Repetir
    CadenaMulti(z)=CadenaMulti(z)+CadenaLineaInicial
    CadenaMulti(z)=CadenaMulti(z)+CadenaLineaIni+Formateo(Multiforma.NoLinea)
                   +CadenaLineaFin
    CadenaMulti(z)=CadenaMulti(z)+CadenaFormPagoIni+Multiforma.CodFormaPago
                    +CadenaFormPagoFin
    CadenaMulti(z)=CadenaMulti(z)+CadenaImporteIni+Format(Multiforma.Importe)+
                    CadenaImporteFin
    CadenaMulti(z)=CadenaMulti(z)+CadenaCambioIni+Format(Multiforma.Cambio)+
                    CadenaCambioFin
    CadenaMulti(z)=CadenaMulti(z)+CadenaValeIni+Multiforma.NoValeAbono+
                    CadenaValeFin
    CadenaMulti(z)=CadenaMulti(z)+CadenaTipoPagoIni+Multiforma.TipoPago+
                    CadenaTipoPagoFin
    CadenaMulti(z)=CadenaMulti(z)+CadenaCobradoIni+Multiforma.Cobrado+
                    CadenaCobradoFin
    CadenaMulti(z)=CadenaMulti(z)+CadenaValeResIni+Multiforma.Abono+
                    CadenaValeResFin
    CadenaMulti(z)=CadenaMulti(z)+'<TipoVale>'+Multiforma.TipoVale+
                    '</TipoVale>'
    CadenaMulti(z)=CadenaMulti(z)+CadenaLineaFinal
    z=z+1
 Hasta siguiente registro Multiforma sea vacío
Si EstaVacio(EncabezadoWeb) entonces
  Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_Abono_Generar');
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
   Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXML('<TPV_Abono_Generar xmlns="http://tempuri.org/">'+
                '<Usuario>'+Usuario+'</Usuario>'+'<Password>'+Password+
                '</password>'+'<NumeroTicket>'+Ticket+'</NumeroTicket>'+
                '<Cliente>'+Cliente+'</Cliente>'+'<CodTienda>'+CodTienda+
                '</CodTienda>'+'<CodDirEnvio>'+CodigoDireccionEnvio+
                '</CodDirEnvio>'+'<EnvioNombre>'+EnvioNombre+'</EnvioNombre>
```

```
                +'<EnvioDireccion>'+EnvioDireccion+'</EnvioDireccion>'+
                +'<EnvioCP>'+EnvioCP+'</EnvioCP>'+'<EnvioPoblacion>'+
                EnvioPoblacion+'</EnvioPoblacion>'+'<EnvioProvincia>'+
                EnvioProvincia+'</EnvioProvincia>'+'<EnvioAtencion>'+
                EnvioAtencion+'</EnvioAtencion>'+'<CodFormaPago>'+
                CodFormaPago+'</CodFormaPago>'+'<CodVendedor>'+CodVendedor
                +'</CodVendedor>'+'<FechaTicket>'+FechaTicket+'</FechaTicket>'+
                '<CodTpv>'+CodTPV+'</CodTpv>'+'<HoraTicket>'+HoraTicket+
                '</HoraTicket>'+'<TotalAPagar>'+TotalAPagar+'</TotalAPagar>'+
                '<ImporteDevolucion>'+ImporteDevolucion+'</ImporteDevolucion>'
                +'<ImporteEntregado>'+CabeceraVenta.ImporteEntregado
                +'</ImporteEntregado>'+'<NumeroTicketAbonado>'+
                NumeroTicketAbonado+'</NumeroTicketAbonado>'+'<IVAIncluido>'+
                TextoIVA+'</IVAIncluido>'+'<Tarifa>'+
                CabeceraVenta.GrupoPrecioIVA+'</Tarifa>'+'<DatosLineasStr>');
ConectorWeb.EscribirCadena(CadenaInicial);
x=1
Repetir
    ConectorWeb.EscribirCadena(Cadena(x))
    x=x+1
Hasta x>i;
ConectorWeb.EscribirCadena(CadenaFinal)
ConectorWeb.EscribirXML('</DatosLineasStr><DatosMultiStr>')
ConectorWeb.EscribirCadena(CadenaInicial)
x=1
Repetir
   ConectorWeb.EscribirCadena(CadenaMulti(x))
   x=x+1
Hasta x>z
ConectorWeb.EscribirCadena(CadenaFinal)
ConectorWeb.EscribirXML('</DatosMultiStr></TPV_Abono_Generar>')
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>'' entonces
   TxtError=EncabezadoWeb.Error
   Salir(Falso)
Si EstaVacio(NodoXml) entonces
    Crear(NodoXml)
NodoXml.cargar(EncabezadoWeb)
NodoXml.guardar('Ticket.xml')
Si ComprobarError(NodoXml,TxtError) entonces
    Salir(Falso)
Salir(Verdadero)
```

Código Original

```
RecTicket.RESET;
RecTicket.SETRANGE(RecTicket."Tipo                    documento",RecTicket."Tipo
documento"::Devolucion);
RecTicket.SETRANGE(RecTicket."Nº documento",Ticket);
IF NOT RecTicket.FINDFIRST() THEN
  ERROR('No existe Información de Ticket');

Conf.RESET;
```

```
Conf.SETRANGE("Cód. tienda", RecTicket."Cód. tienda");
Conf.SETRANGE("Cód. TPV", RecTicket."Cód. TPV");
IF Conf.FINDFIRST() THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
   END
ELSE
  ERROR('No Existe Configuración TPV');

FechaTicket        :=FORMAT(RecTicket."Fecha documento");
CodTPV             :=RecTicket."Cód. TPV";
HoraTicket         :=FORMAT(RecTicket."Hora documento");
TotalApagar        :=FORMAT(RecTicket."Total a pagar");
ImporteEntregado   :=FORMAT(RecTicket."Importe entregado");
ImporteVenta       :=FORMAT(RecTicket."Total venta provisional");

 BEGIN
    IF ISCLEAR(XMLDom) THEN
      CREATE (XMLDom);

    CadenaInicial:='<NewDataSet>';
    CadenaLineaInicial:='<Lineas>';
    CadenaFinal:='</NewDataSet>';
    CadenaLineaFinal:='</Lineas>';

    CadenaTipoIni:='<Tipo>';
    CadenaTipoFin:='</Tipo>';


    CadenaNumeroIni:='<No.>';
    CadenaNumeroFin:='</No.>';

    CadenaQtyIni:='<Quantity>';
    CadenaQtyFin:='</Quantity>';

    CadenaUnitPriceIni:='<UnitPrice>';
    CadenaUnitPriceFin:='</UnitPrice>';

    CadenaLineDisIni:='<LineDiscount>';
    CadenaLineDisFin:='</LineDiscount>';
    CadenaDescriptionIni := '<Description>';
    CadenaDescriptionFin := '</Description>';

    CadenaLineaini    :='<Linea>';
    CadenaLineaFIni   :='</Linea>';
    CadenaFormPagoIni :='<FormPago>';
    CadenaFormPagoFin :='</FormPago>';
    CadenaImporteIni  :='<Importe>';
    CadenaImporteFin  :='</Importe>';
    CadenaCambioIni   :='<Cambio>';
    CadenaCambioFin   :='</Cambio>';
```

```
    CadenaValeIni     :='<Vale>';
    CadenaValeFin     :='</Vale>';
    CadenaTipoPagoIni :='<TipoPago>';
    CadenaTipoPagoFin :='</TipoPago>';
    CadenaCobradoIni  :='<Cobrado>';
    CadenaCobradoFin  :='</Cobrado>';
    CadenaValeResIni  :='<ValeRes>';
    CadenaValeResFin  :='</ValeRes>';

    PsalesHeader.RESET;
    PsalesHeader.SETCURRENTKEY("Pre-Assigned No.");
    PsalesHeader.SETRANGE(PsalesHeader."Pre-Assigned No.",Ticket);
    IF PsalesHeader.FINDFIRST() THEN
      NumeroDOC:=PsalesHeader."No."
    ELSE
      ERROR('ERROR no encuentro la cabecera de Abono en el histórico');
    TextoIva := FORMAT(PsalesHeader."Prices Including VAT");

    PsalesLine.RESET;
    PsalesLine.SETRANGE(PsalesLine."Document No.",NumeroDOC);
    IF PsalesLine.FINDSET() THEN
    BEGIN
      i:=1;
     REPEAT
        Cadena[i] := Cadena[i] + CadenaLineaInicial;
        Cadena[i] := Cadena[i] + CadenaTipoIni+FORMAT(PsalesLine.Type)+ CadenaTipoFin;
        Cadena[i] := Cadena[i] + CadenaNumeroIni+PsalesLine."No."+ CadenaNumeroFin;
        Cadena[i] := Cadena[i] +
CadenaQtyIni+FORMAT(PsalesLine.Quantity)+CadenaQtyFin;
        Cadena[i] := Cadena[i] + CadenaUnitPriceIni+FORMAT(PsalesLine."Unit
Price")+CadenaUnitPriceFin;
        Cadena[i] := Cadena[i] + CadenaLineDisIni+FORMAT(PsalesLine."Line Discount
%")+CadenaLineDisFin;

        tempDesc:=PsalesLine.Description;
        ConvertDesc:='';
        fin:=FALSE;
        WHILE NOT fin DO
        BEGIN
          iPos:=STRPOS(tempDesc,'&');
          IF  iPos = 0 THEN
          BEGIN
            ConvertDesc+=tempDesc;
            fin:=TRUE;
          END
          ELSE
          BEGIN
            ConvertDesc += COPYSTR(tempDesc,1,iPos-1) + '&amp;';
            tempDesc:= COPYSTR(tempDesc,iPos+1);
          END;
        END;
        Cadena[i] := Cadena[i] + CadenaDescriptionIni + ConvertDesc + CadenaDescriptionFin;
        Cadena[i] := Cadena[i] + CadenaLineaFinal;
        i:=i+1;
```

```
    UNTIL PsalesLine.NEXT=0;
  END;

  Multiforma.RESET;
  Multiforma.SETRANGE(Multiforma."Tipo Documento",Multiforma."Tipo Documento"::Devolucion);
  Multiforma.SETRANGE(Multiforma."Nº Documento",Ticket);
  z:=1;

  IF Multiforma.FINDSET() THEN REPEAT
    CadenaMulti[z] := CadenaMulti[z] + CadenaLineaInicial;
    CadenaMulti[z] := CadenaMulti[z] + CadenaLineaini+FORMAT(Multiforma."Nº Linea") +CadenaLineaFIni;
    CadenaMulti[z] := CadenaMulti[z] + CadenaFormPagoIni+Multiforma."Cod. Forma Pago" + CadenaFormPagoFin;
    CadenaMulti[z] := CadenaMulti[z] + CadenaImporteIni+FORMAT(Multiforma.Importe) +CadenaImporteFin;
    CadenaMulti[z] := CadenaMulti[z] + CadenaCambioIni+FORMAT(Multiforma.Cambio) +CadenaCambioFin;
    CadenaMulti[z] := CadenaMulti[z] + CadenaValeIni+Multiforma."Nº vale de abono"+CadenaValeFin;
    CadenaMulti[z] := CadenaMulti[z] + CadenaTipoPagoIni+FORMAT(Multiforma."Tipo pago") +CadenaTipoPagoFin;
    CadenaMulti[z] := CadenaMulti[z] + CadenaCobradoIni+FORMAT(Multiforma.Cobrado) +CadenaCobradoFin;
    CadenaMulti[z] := CadenaMulti[z] + CadenaValeResIni+Multiforma."Nº bono / Reserva / Abono"+CadenaValeResFin;
    CadenaMulti[z] := CadenaMulti[z] + '<TipoVale>' + FORMAT(Multiforma."Tipo de vale") + '</TipoVale>';
    CadenaMulti[z] := CadenaMulti[z] + CadenaLineaFinal;
    z:=z+1;
  UNTIL Multiforma.NEXT=0;

  IF ISCLEAR(SoapHttpConn) THEN
    CREATE(SoapHttpConn);

  SoapHttpConn.Property('EndPointURL', URLWS);
  SoapHttpConn.Connect;

  SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_Abono_Generar');
  SoapHttpConn.BeginMessage;
  IF ISCLEAR(SoapSerialize) THEN
    CREATE(SoapSerialize);
  SoapSerialize.Init(SoapHttpConn.InputStream);
  SoapSerialize.StartEnvelope('','STANDARD','');
  SoapSerialize.StartBody('');
  SoapSerialize.WriteXML('<TPV_Abono_Generar xmlns="http://tempuri.org/">'+
                         '<Usuario>'+Usuario+'</Usuario>'+
                         '<password>'+Password+'</password>'+
                         '<NumeroTicket>'+Ticket+'</NumeroTicket>'+
                         '<Cliente>'+Cliente+'</Cliente>'+
                         '<CodTienda>'+CodTienda+'</CodTienda>'+
                         '<CodDirEnvio>'+ CodigoDireccionEnvio +'</CodDirEnvio>'+
                         '<EnvioNombre>'+EnvioNombre+'</EnvioNombre>'+
```

```
                    '<EnvioDireccion>'+EnvioDireccion+'</EnvioDireccion>'+
                    '<EnvioCP>'+EnvioCP+'</EnvioCP>'+
                    '<EnvioPoblacion>'+EnvioPoblacion+'</EnvioPoblacion>'+
                    '<EnvioProvincia>'+EnvioProvincia+'</EnvioProvincia>'+
                    '<EnvioAtencion>'+EnvioAtencion+'</EnvioAtencion>'+
                    '<CodFormaPago>'+CodFormaPago+'</CodFormaPago>'+
                    '<CodVendedor>'+CodVendedor+'</CodVendedor>'+
                    '<FechaTicket>'+FechaTicket+'</FechaTicket>'+
                    '<CodTPV>'+ CodTPV +'</CodTPV>'+
                    '<HoraTicket>'+HoraTicket+'</HoraTicket>'+
                    '<TotalApagar>'+TotalApagar+'</TotalApagar>'+
                    '<ImporteDevolucion>'+ImpDevolucion+'</ImporteDevolucion>'+
                    '<ImporteEntregado>'+FORMAT(PsalesHeader."Importe
Entregado")+'</ImporteEntregado>'+
                    '<NumeroTicketAbonado>'+
NumeroTicketAbonado+'</NumeroTicketAbonado>'+
                    '<IvaIncluido>' + TextoIva + '</IvaIncluido>' +
                    '<Tarifa>' + PsalesHeader."Customer Price Group" + '</Tarifa>' +
                    '<DatosLineasStr>');

    SoapSerialize.WriteString(CadenaInicial);
    x:=1;
    REPEAT
      SoapSerialize.WriteString(Cadena[x]);
      x:=x+1;
    UNTIL x > i;
    SoapSerialize.WriteString(CadenaFinal);
    SoapSerialize.WriteXML('</DatosLineasStr><DatosMultiStr>');
    SoapSerialize.WriteString(CadenaInicial);
    x:=1;
    REPEAT
      SoapSerialize.WriteString(CadenaMulti[x]);
      x:=x+1;
    UNTIL x > z;
    SoapSerialize.WriteString(CadenaFinal);
    SoapSerialize.WriteXML('</DatosMultiStr></TPV_Abono_Generar>');
    SoapSerialize.EndBody;
    SoapSerialize.EndEnvelope;
    SoapHttpConn.EndMessage;

    IF SoapHttpConn.Error <> '' THEN BEGIN
       TxtError:=SoapHttpConn.Error;
       EXIT(FALSE);
    END;

    IF ISCLEAR(XMLDom) THEN
       CREATE(XMLDom);

    XMLDom.async := FALSE;
    XMLDom.load(SoapHttpConn.OutputStream);
     XMLDom.save(cduILR.DameDirXML() + 'Ticket.xml');

    IF ComprobarERROR(XMLDom,TxtError) THEN BEGIN
      EXIT(FALSE);
```

```
    END;
    EXIT(TRUE);
END;
```

TPV_Reserva_Generar(Usuario,Password,Ticket,Envio,Cliente,EnvioNombre, EnvioDireccion,EnvioCP,EnvioPoblacion,EnvioProvincia,EnvioAtencion, CodTienda,CodFormaPago,CodVendedor,NumTicketAbonado,ImpAnticipo);

Esta función sirve para mandar la reserva con todos los datos (cabecera, líneas y formas de pago) a CENTRAL. Entonces, esta función se invoca una vez que el sistema ha registrado, de forma local, la reserva y así poder enviarla a CENTRAL y que los datos queden centralizados.

PseudoCódigo

```
TablaTicket se inicializa
TablaTicket se filtra TipoDocumento sea Reserva
TablaTicket se filtra NoDocumento sea Ticket
Si no encuentra registros TablaTicket entonces
   ERROR('No existe información del Ticket');
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
CabeceraVenta se inicializa
CabeceraVenta se filtra TipoDocumento sea Pedido
CabeceraVenta se filtra No sea Ticket
Si encuenta registros CabeceraVenta entonces
   NumeroDoc=CabeceraVenta.No
Sino
  ERROR('ERROR no encuentro la cabecera de la reserva');
TextoIVA=Formateo(CabeceraVenta.GrupoPrecioIVA)
FechaTicket=Formatear(TablaTicket.FechaDocumento);
CodTpv=TablaTicket.CodTpv
HoraTicket=Formatear(TablaTicket.HoraDocumento)
TotalaPagar=TablaTicket.TotalaPagar
ImporteEntregado=Formatear(ImpAnticipo)
ImporteDevolucion=Formatear(TablaTicket.TotalDevolucionRegistrada)

CadenaInicial:='<NewDataSet>';
CadenaLineaInicial:='<Lineas>';
CadenaFinal:='</NewDataSet>';
CadenaLineaFinal:='</Lineas>';
CadenaNumeroIni:='<No.>';
CadenaNumeroFin:='</No.>';

CadenaQtyIni:='<Quantity>';
```

```
CadenaQtyFin:='</Quantity>';
CadenaUnitPriceIni:='<UnitPrice>';
CadenaUnitPriceFin:='</UnitPrice>';
CadenaLineDisIni:='<LineDiscount>';
CadenaLineDisFin:='</LineDiscount>';
CabeceraVenta se inicializa
CabeceraVenta se filtra TipoDocumento sea Pedido
CabeceraVenta se filtra NoDocumento sea NumeroDoc
Si encuentra registros CabeceraVenta entonces
   i=1
   Repetir
      Cadena(i)=Cadena(i)+CadenaLineaInicial
      Cadena(i)=Cadena(i)+CadenaNumeroIni+LineasVenta.No+CadenaNumeroFin
      Cadena(i)=Cadena(i)+CadenaQtyIni+Formateo(LineasVenta.Cantidad)+
                CadenaQtyFin
      Cadena(i)=Cadena(i)+CadenaPrecioIni+Formateo(LineasVenta.Precio)+
                CadenaPrecioFin
      Cadena(i)=Cadena(i)+CadenaDescIni+Formateo(LineasVenta.Descuento)+
                CadenaDescFin
      Cadena(i)=Cadena(i)+CadenaLineaFinal
      i=i+1
   Hasta siguiente registro LineasVenta sea vacio
Si EstaVacio(EncabezadoWeb) entonces
   Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_Reserva_Generar');
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXML('<TPV_Reserva_Generar xmlns="http://tempuri.org/">'+
                '<Usuario>'+Usuario+'</Usuario>'+'<Password>'+Password+
                '</password>'+'<NumeroTicket>'+Ticket+'</NumeroTicket>'+
                '<Cliente>'+Cliente+'</Cliente>'+'<CodTienda>'+CodTienda+
                '</CodTienda>'+'<CodDirEnvio>'+CodigoDireccionEnvio+
                '</CodDirEnvio>'+'<EnvioNombre>'+EnvioNombre+'</EnvioNombre>
                +'<EnvioDireccion>'+EnvioDireccion+'</EnvioDireccion>'+
                +'<EnvioCP>'+EnvioCP+'</EnvioCP>'+'<EnvioPoblacion>'+
                EnvioPoblacion+'</EnvioPoblacion>'+'<EnvioProvincia>'+
                EnvioProvincia+'</EnvioProvincia>'+'<EnvioAtencion>'+
                EnvioAtencion+'</EnvioAtencion>'+'<CodFormaPago>'+
                CodFormaPago+'</CodFormaPago>'+'<CodVendedor>'+CodVendedor
                +'</CodVendedor>'+'<FechaTicket>'+FechaTicket+'</FechaTicket>'+
                '<CodTpv>'+CodTPV+'</CodTpv>'+'<HoraTicket>'+HoraTicket+
                '</HoraTicket>'+'<TotalAPagar>'+TotalAPagar+'</TotalAPagar>'+
                '<Recibido>'+Recibido+'</Recibido>'+'<Cambio>'+Cambio+
                '</Cambio>'+'<CodFormaPago2>'+CodFormaPago2+
               '</CodFormaPago2>'+'<ImporteDevolucion>'+ImporteDevolucion
               +'</ImporteDevolucion>' +'<ImporteEntregado>'
               +CabeceraVenta.ImporteEntregado+'</ImporteEntregado>'+
               '<NumeroTicketAbonado>'+NumeroTicketAbonado
```

```
                +'</NumeroTicketAbonado>'+'<IVAIncluido>'+
                 TextoIVA+'</IVAIncluido>'+'<Tarifa>'+
                 CabeceraVenta.GrupoPrecioIVA+'</Tarifa>'+'<DatosLineasStr>');
ConectorWeb.EscribirCadena(CadenaInicial);
x=1
Repetir
    ConectorWeb.EscribirCadena(Cadena(x))
    x=x+1
Hasta x>i;
ConectorWeb.EscribirCadena(CadenaFinal)
ConectorWeb.EscribirXML('</DatosLineasStr></TPV_Abono_Generar>')
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>'' entonces
   TxtError=EncabezadoWeb.Error
   Salir(Falso)
Si EstaVacio(NodoXml) entonces
    Crear(NodoXml)
NodoXml.cargar(EncabezadoWeb)
NodoXml.guardar('Ticket.xml')
Si ComprobarError(NodoXml,TxtError) entonces
    Salir(Falso)
Salir(Verdadero)
```

Código Original

```
RecTicket.RESET;
RecTicket.SETRANGE(RecTicket."Tipo documento",RecTicket."Tipo documento"::Reserva);
RecTicket.SETRANGE(RecTicket."Nº documento",Ticket);
IF NOT RecTicket.FINDFIRST() THEN
  ERROR('No existe Información de Ticket');

Conf.RESET;
Conf.SETRANGE("Cód. tienda", RecTicket."Cód. tienda");
Conf.SETRANGE("Cód. TPV", RecTicket."Cód. TPV");
IF Conf.FINDFIRST() THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
   END
ELSE
  ERROR('No Existe Configuración TPV');

PsalesHeader.RESET;
PsalesHeader.SETRANGE(PsalesHeader."Document Type",PsalesHeader."Document
Type"::Order);
PsalesHeader.SETRANGE(PsalesHeader."No.",Ticket);
IF PsalesHeader.FINDFIRST() THEN
  NumeroDOC:=PsalesHeader."No."
ELSE
  ERROR('ERROR no encuentro la cabecera de Reserva');
```

```
TextoIva := FORMAT(PsalesHeader."Prices Including VAT");
FechaTicket        :=FORMAT(RecTicket."Fecha documento");
CodTPV             :=RecTicket."Cód. TPV";
HoraTicket         :=FORMAT(RecTicket."Hora documento");
TotalApagar        :=FORMAT(RecTicket."Total a pagar");
IMporteDevolucion  :=FORMAT(RecTicket."Total devolución registrada");
ImporteEntregado   :=FORMAT(ImpAnticipo);

 BEGIN
   IF ISCLEAR(XMLDom) THEN
     CREATE (XMLDom);

   CadenaInicial:='<NewDataSet>';
   CadenaLineaInicial:='<Lineas>';
   CadenaFinal:='</NewDataSet>';
   CadenaLineaFinal:='</Lineas>';

   CadenaNumeroIni:='<No.>';
   CadenaNumeroFin:='</No.>';

   CadenaQtyIni:='<Quantity>';
   CadenaQtyFin:='</Quantity>';

   CadenaUnitPriceIni:='<UnitPrice>';
   CadenaUnitPriceFin:='</UnitPrice>';

   CadenaLineDisIni:='<LineDiscount>';
   CadenaLineDisFin:='</LineDiscount>';

   PsalesLine.RESET;
   PsalesLine.SETRANGE(PsalesLine."Document Type",PsalesLine."Document
Type"::Order);
   PsalesLine.SETRANGE(PsalesLine."Document No.",NumeroDOC);
   IF PsalesLine.FINDSET() THEN
   BEGIN
     i:=1;
    REPEAT
       Cadena[i] := Cadena[i] + CadenaLineaInicial;
       Cadena[i] := Cadena[i] + CadenaNumeroIni+PsalesLine."No."+ CadenaNumeroFin;
       Cadena[i] := Cadena[i] +
CadenaQtyIni+FORMAT(PsalesLine.Quantity)+CadenaQtyFin;
       Cadena[i] := Cadena[i] + CadenaUnitPriceIni+FORMAT(PsalesLine."Unit
Price")+CadenaUnitPriceFin;
       Cadena[i] := Cadena[i] + CadenaLineDisIni+FORMAT(PsalesLine."Line Discount
%")+CadenaLineDisFin;
       Cadena[i] := Cadena[i] + CadenaLineaFinal;
       i:=i+1;
    UNTIL PsalesLine.NEXT=0;
   END;
   IF ISCLEAR(SoapHttpConn) THEN
     CREATE(SoapHttpConn);

   SoapHttpConn.Property('EndPointURL', URLWS);
   SoapHttpConn.Connect;
```

```
SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_Reserva_Generar');
SoapHttpConn.BeginMessage;
IF ISCLEAR(SoapSerialize) THEN
  CREATE(SoapSerialize);
SoapSerialize.Init(SoapHttpConn.InputStream);
SoapSerialize.StartEnvelope('','STANDARD','');
SoapSerialize.StartBody('');
SoapSerialize.WriteXML('<TPV_Reserva_Generar xmlns="http://tempuri.org/">'+
                '<Usuario>'+Usuario+'</Usuario>'+
                '<password>'+Password+'</password>'+
                '<NumeroTicket>'+Ticket+'</NumeroTicket>'+
                '<Cliente>'+Cliente+'</Cliente>'+
                '<CodTienda>'+CodTienda+'</CodTienda>'+
                '<CodDirEnvio>'+ CodigoDireccionEnvio +'</CodDirEnvio>'+
                '<EnvioNombre>'+EnvioNombre+'</EnvioNombre>'+
                '<EnvioDireccion>'+EnvioDireccion+'</EnvioDireccion>'+
                '<EnvioCP>'+EnvioCP+'</EnvioCP>'+
                '<EnvioPoblacion>'+EnvioPoblacion+'</EnvioPoblacion>'+
                '<EnvioProvincia>'+EnvioProvincia+'</EnvioProvincia>'+
                '<EnvioAtencion>'+EnvioAtencion+'</EnvioAtencion>'+
                '<CodFormaPago>'+CodFormaPago+'</CodFormaPago>'+
                '<CodVendedor>'+CodVendedor+'</CodVendedor>'+
                '<FechaTicket>'+FechaTicket+'</FechaTicket>'+
                '<CodTPV>'+ CodTPV +'</CodTPV>'+
                '<HoraTicket>'+HoraTicket+'</HoraTicket>'+
                '<TotalApagar>'+TotalApagar+'</TotalApagar>'+
                '<Recibido>'+Recibido+'</Recibido>'+
                '<Cambio>'+Cambio+'</Cambio>'+
                '<CodFormaPago2>'+CodFormaPago2+'</CodFormaPago2>'+
               '<IMporteDevolucion>'+IMporteDevolucion+'</IMporteDevolucion>'+
                '<ImporteEntregado>'+ImporteEntregado+'</ImporteEntregado>'+
                '<Vale>'+Vale+'</Vale>'+
                '<IvaIncluido>' + TextoIva + '</IvaIncluido>' +
                '<Tarifa>' + PsalesHeader."Customer Price Group" + '</Tarifa>' +
                '<NumBono>' + NumTicketAbonado + '</NumBono>' +
                '<DatosLineasStr>');
SoapSerialize.WriteString(CadenaInicial);
x:=1;
REPEAT
  SoapSerialize.WriteString(Cadena[x]);
  x:=x+1;
UNTIL x > i;
SoapSerialize.WriteString(CadenaFinal);
SoapSerialize.WriteXML('</DatosLineasStr></TPV_Reserva_Generar>');
SoapSerialize.EndBody;
SoapSerialize.EndEnvelope;
SoapHttpConn.EndMessage;

IF SoapHttpConn.Error <> '' THEN
  EXIT(FALSE);

IF ISCLEAR(XMLDom) THEN
  CREATE(XMLDom);
```

```
    XMLDom.async := FALSE;
    XMLDom.load(SoapHttpConn.OutputStream);
    XMLDom.save(cduILR.DameDirXML() + 'Ticket.xml');

    IF ComprobarERROR(XMLDom,TxtError) THEN BEGIN
      EXIT(FALSE);
    END;
    EXIT(TRUE);
END;
```

TPV_Reserva_Cancelar(Usuario,Password,Ticket,FechaReg,CodTpv)

Esta función se invoca para enviar la cancelación de la reserva a CENTRAL y que el estado de las reservas esté actualizado al momento. Además de realizar la cancelación, se debe dar de alta un bono, con número idéntico al de la reserva, ya que por defecto, cuando se cancela una reserva se emite un bono que se puede canjear en las próximas compras.

PseudoCódigo

```
CabeceraVenta se inicializa
CabeceraVenta se filtra TipoDocumento sea Pedido
CabeceraVenta se filtra No sea Ticket
Si encuentra registro CabeceraVenta entonces
    NumDoc=CabeceraVenta.No
Sino
    ERROR('ERROR no encuentro la cabecera de la reserva')
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
       UrlWs:= ConfigTpv.URLCentral
   Sino
       ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si EstaVacio(NodoXml) entonces
   Crear(NodoXml)
Si EncabezadoWeb.Vacio entonces
    Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_Reserva_Cancelar);
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXml('<TPV_Reserva_Cancelar xmlns=http://tempuri.org/>'+
                '<Usuario>'+Usuario+'</Usuario>'+'<password>'+Password+
                '</password>'+'<Ticket>'+Ticket+'</Ticket>'+'<CodTienda>'
```

```
                CabeceraVenta.CodTienda+'</CodTienda>+'<FechaRegistro>'+
                FechaReg+'</FechaRegistro>'+'<CodTpv>'+CodTpv+'</CodTpv>')
ConectorWeb.EscribirXml('</TPV_Reserva_Cancelar>')
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>'' entonces
   Salir(Falso)
Si EstaVacio(NodoXml) entonces
   Crear(NodoXml)
NodoXml.cargar(EncabezadoWeb)
NodoXml.guardar('Ticket.xml')
Si ComprobarERROR(NodoXml,txtError) entonces
    Salir(Falso)
Salir(verdadero)
```

Código Original

```
PsalesHeader.RESET;
PsalesHeader.SETRANGE(PsalesHeader."Document Type",PsalesHeader."Document
Type"::Order);
PsalesHeader.SETRANGE(PsalesHeader."No.",Ticket);
IF PsalesHeader.FINDFIRST() THEN
  NumeroDOC:=PsalesHeader."No."
ELSE
  ERROR('ERROR no encuentro la cabecera de Reserva');

Conf.RESET;
Conf.SETRANGE("Cód. tienda", PsalesHeader."Cód. tienda");
Conf.SETRANGE("Cód. TPV", PsalesHeader."Cód. TPV");
IF Conf.FINDFIRST() THEN
   BEGIN
     IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
       URLWS:=Conf."URL Servicio WEB CENTRAL"
     ELSE
       ERROR('No existe Configuración del Servicio WEB');
    END
ELSE
   ERROR('No Existe Configuración TPV');
  BEGIN
     IF ISCLEAR(XMLDom) THEN
       CREATE (XMLDom);

     IF ISCLEAR(SoapHttpConn) THEN
       CREATE(SoapHttpConn);

     SoapHttpConn.Property('EndPointURL', URLWS);
     SoapHttpConn.Connect;

     SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_Reserva_Cancelar');
     SoapHttpConn.BeginMessage;
     IF ISCLEAR(SoapSerialize) THEN
       CREATE(SoapSerialize);
     SoapSerialize.Init(SoapHttpConn.InputStream);
     SoapSerialize.StartEnvelope('','STANDARD','');
```

```
    SoapSerialize.StartBody('');
    SoapSerialize.WriteXML('<TPV_Reserva_Cancelar xmlns="http://tempuri.org/">'+
                    '<Usuario>'+Usuario+'</Usuario>'+
                    '<password>'+Password+'</password>'+
                    '<Ticket>'+Ticket+'</Ticket>'+
                    '<CodTienda>'+ PsalesHeader."Cód. tienda"+'</CodTienda>'+
                    '<FechaRegistro>'+ FORMAT(FechaReg)+'</FechaRegistro>'+
                    '<CodTPV>'+ FORMAT(CodTPV)+'</CodTPV>');
    SoapSerialize.WriteXML('</TPV_Reserva_Cancelar>');
    SoapSerialize.EndBody;
    SoapSerialize.EndEnvelope;
    SoapHttpConn.EndMessage;

    IF SoapHttpConn.Error <> '' THEN
      EXIT(FALSE);

    IF ISCLEAR(XMLDom) THEN
      CREATE(XMLDom);

    XMLDom.async := FALSE;
    XMLDom.load(SoapHttpConn.OutputStream);
    XMLDom.save(cduILR.DameDirXML() + 'Ticket.xml');

    IF ComprobarERROR(XMLDom,TxtError) THEN BEGIN
      EXIT(FALSE);
    END;
    EXIT(TRUE);
END;
```

TPV_Consulta_Stock(Usuario,Password,Producto,AlmTMP)

Esta función devuelve, en una matriz, el stock disponible por cada almacén del producto que se ha pasado por parámetro. A continuación, el sistema muestra esa matriz por pantalla.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si EstaVacio(NodoXml) entonces
   Crear(NodoXml)
CadenaInicial='<NewDataSet>'
CadenaLineaInicial='<Lineas>'
CadenaFinal='</NewDataSet>'
CadenaLineaFinal='</Lineas>'
CadenaNumeroIni='<No.>'
CadenaNumeroFin='</No.>'
```

```
CadenaQtyIni='<Alma>'
CadenaQtyFin='</Alma>'
TablaAlmacen se inicializa
Si no encuentra registros TablaAlmacen entonces
   ERROR('No hay almacenes creados')
i=1
Repetir
   Cadena(i)=Cadena(i)+CadenaLineaInicial
   Cadena(i)=Cadena(i)+CadenaQtyIni+TablaAlmacen.Codigo+CadenaQtyFin
   Cadena(i)=Cadena(i)+CadenaLineaFinal
   i=i+1
Hasta siguiente registro TablaAlmacen sea vacío
Si EstaVacío(EncabezadoWeb) entonces
   Crear(EncabezadoWeb);
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_Consulta_Stock');
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXml('<TPV_Consulta_Stock xmlns=http://tempuri.org/>'+
                  '<Usuario>'+Usuario+'</Usuario>'+'<password>'+Password+
                 '</password>'+'<NumeroProducto>'+Producto+'</NumeroProducto>'+
                 '<DatosLineasStr>')
ConectorWeb.EscribirCadena(CadenaInicial)
x=1
Repetir
   ConectorWeb.EscribirCadena(Cadena(x))
   x=x+1
Hasta x>i
ConectorWeb.EscribirCadena(CadenaFinal)
ConectorWeb.EscribirXml('</DatosLineasStr></TPV_Consulta_Stock>')
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>'' entonces
    Salir(Falso)
Si EstaVacío(NodoXml) entonces
   Crear(NodoXml)
NodoXml.Cargar(EncabezadoWeb)
NodoXml.guardar('ConsultaStock.xml')
NodoLista=NodoXml.CogerElementos(Almacen)
Si NodoLista.Longitud=0 entonces
   Salir(Falso)
Si ComprobarError(NodoXml,txtError) entonces
    Salir(Falso)
Sino
   i=0
  TablaAlmacen se inicializa
  TablaAlmacen se posiciona primer registro
  AlmaTMP.Copiar(TablaAlmacen);
```

```
  Repetir
     NodoLista=NodoXml.CogerElementos(Almacen)
     CodAlma=NodoLista.elemento(i).texto
     TablaAlmacen se inicializa
     TablaAlmacen se filtra Codigo sea CodAlma
     Si encuentra registro TablaAlmacen entonces
        NodoLista=NodoXml.CogerElementos(Existencias)
        Evaluar(Exis,NodoLista.elemento(i).texto)
        TablaAlmacen.Existencias=Exis
        Modificar registro TablaAlmacen
  Hasta i=NodoLista.Longitud
Salir(Verdadero)
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF Conf.FINDFIRST() THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
   END
ELSE
  ERROR('No Existe Configuración TPV');
 BEGIN
    IF ISCLEAR(XMLDom) THEN
      CREATE (XMLDom);
    CadenaInicial:='<NewDataSet>';
    CadenaLineaInicial:='<Lineas>';
    CadenaFinal:='</NewDataSet>';
    CadenaLineaFinal:='</Lineas>';

    CadenaNumeroIni:='<No.>';
    CadenaNumeroFin:='</No.>';
    CadenaQtyIni:='<Alma>';
    CadenaQtyFin:='</Alma>';
    Alma.RESET ;
    IF NOT Alma.FINDFIRST() THEN
      ERROR('No hay creados almacenes');
    BEGIN
      i:=1;
     REPEAT
        Cadena[i] := Cadena[i] + CadenaLineaInicial;
        Cadena[i] := Cadena[i] + CadenaQtyIni+Alma.Code+CadenaQtyFin;
        Cadena[i] := Cadena[i] + CadenaLineaFinal;
        i:=i+1;
     UNTIL Alma.NEXT=0;
    END;
    IF ISCLEAR(SoapHttpConn) THEN
      CREATE(SoapHttpConn);
    SoapHttpConn.Property('EndPointURL', URLWS);
    SoapHttpConn.Connect;
    SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_Consulta_Stock');
```

```
SoapHttpConn.BeginMessage;
IF ISCLEAR(SoapSerialize) THEN
  CREATE(SoapSerialize);
SoapSerialize.Init(SoapHttpConn.InputStream);
SoapSerialize.StartEnvelope('','STANDARD','');
SoapSerialize.StartBody('');
SoapSerialize.WriteXML('<TPV_Consulta_Stock xmlns="http://tempuri.org/">'+
                '<Usuario>'+Usuario+'</Usuario>'+
                '<password>'+Password+'</password>'+
                '<NumeroProducto>'+Producto+'</NumeroProducto>'+
                '<DatosLineasStr>');
SoapSerialize.WriteString(CadenaInicial);
x:=1;
REPEAT
 SoapSerialize.WriteString(Cadena[x]);
 x:=x+1;
UNTIL x > i;
SoapSerialize.WriteString(CadenaFinal);
SoapSerialize.WriteXML('</DatosLineasStr></TPV_Consulta_Stock>');
SoapSerialize.EndBody;
SoapSerialize.EndEnvelope;
SoapHttpConn.EndMessage;

IF SoapHttpConn.Error <> '' THEN
  EXIT(FALSE);

IF ISCLEAR(XMLDom) THEN
  CREATE(XMLDom);

XMLDom.async := FALSE;
XMLDom.load(SoapHttpConn.OutputStream);
XMLDom.save(cduILR.DameDirXML() + 'ConsultaStock.xml');
DOMNodelist:= XMLDom.getElementsByTagName('Almacen');
IF DOMNodelist.length = 0 THEN
 EXIT(FALSE);

IF ComprobarERROR(XMLDom,TxtError) THEN BEGIN
 EXIT(FALSE);
END
ELSE
 BEGIN
 i:=0;
  Alma.RESET ;
  Alma.FIND('-');
  AlmaTMP.COPY(Alma);
   REPEAT
     DOMNodelist:= XMLDom.getElementsByTagName('Almacen');
     CodAlma:=DOMNodelist.item(i).text;
     Alma.RESET;
     Alma.SETRANGE(Alma.Code,CodAlma);
     IF Alma.FIND('-') THEN BEGIN
        DOMNodelist:= XMLDom.getElementsByTagName('Existencias');
        EVALUATE(Exis,DOMNodelist.item(i).text);
        Alma.Existencias:=Exis;
```

```
            Alma.MODIFY;
          END;
        i:=i+1;
       UNTIL i= DOMNodelist.length;
     END;
    EXIT(TRUE);
END;
```

TPV_RegDiario(Usuario,Password,FechaRegistro,TipoDocumento,NºDocumento, TipoMovimiento,NºCuenta,VDescripcion,VImporte,TipoContraPartida, NºContraPartida,CodTienda,CodTpv,NumTurno,TipoMovTraspaso,Signo)

Esta función se invoca desde varios sitios y sirve para hacer movimiento entre distintas cuentas, dependiendo de desde dónde se invoque: Uno de ellos es cuando se registra un gasto del día. Entonces, el sistema invoca la función para que en CENTRAL quede constancia de ese gasto y de los movimientos de cuenta pertinentes. Otro sitio donde se invoca es cuando se cierra caja del día, que realiza el movimiento de saldo de la caja.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
FechaText=Formateo(FechaRegistro)
TipoDocTex=Formateo(TipoDocumento)
TipoMovTex=Formateo(TipoMovimiento)
TipoMovTrasTex=Formateo(TipoMovTraspaso)
ImporteTex=Formateo(VImporte)
NumTurnoTex=Formateo(NumTurno)
TipoContrapTex=Formateo(TipoContrapartida)
Si EstaVacío(NodoXml) entonces
   Crear(NodoXml)
Si EncabezadoWeb.Vacio entonces
    Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_RegDiario);
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXml('<TPV_RegDiario xmlns=http://tempuri.org/>'+
```

```
                ‘<Usuario>’+Usuario+’</Usuario>’+’<password>’+Password+
                ‘</Password>’+’<FechaRegistro>’+FechaText+’</FechaRegistro>’+
                ‘<TipoDocumento>’+TipoDocTex+’</TipoDocumento>’+
                ‘<NDocumento>’+NoDocumento+’</NDocumento>’+
                ‘<TipoMovimiento>’+TipoMovTex+’</TipoMovimiento>’+
                ’<NCuenta>’+NoCuenta+’</NCuenta>’+’<VDescripcion>’+
                VDescripcion+’</VDescripcion>’+’<VImporteTex>’+ImporteTex+
                ‘</VImporteTex>’+’<TipoContrapartidaTex>’+TipoContraptex+
                ‘</TipoContrapartidaTex>’+’<NContrapartida>’+NoContrapartida+
                ‘</NContrapartida>’+’<CodTienda>’+CodTienda+’</CodTienda>’+
                ‘<CodTpv>’+CodTpv+’</CodTpv>’+’<NTurnoTex>’+NumTurnoTex+
                ‘</NumTurnoTex>’+’<TipoMovTraspasoTex>’+TipoMovTraspTex+
                ‘</TipoMovTraspTex>’+’<Signo>’+Signo+’</Signo>’)
ConectorWeb.EscribirXml(‘</TPV_RegDiario>’)
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>’’ entonces
    Salir(FALSO)
Si EstaVacío(NodoXml) entonces
   Crear(NodoXml)
NodoXml.Cargar(EncabezadoWeb)
NodoXml.Guardar(‘Diario.xml’)
Si ComprobarERROR(NodoXml,txtError) entonces
   Salir(Falso)
Salir(Verdadero)
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. tienda", CodTienda);
Conf.SETFILTER("Cód. TPV", CodTPV);
IF Conf.FINDFIRST() THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
   END
ELSE
  ERROR('No Existe Configuración TPV');

FechaText         := FORMAT(FechaRegistro);
TipoDocTex        := FORMAT(TipoDocumento);
TipoMovTex        := FORMAT(TipoMovimiento);
TipoMovTraspTex   := FORMAT(TipoMovTraspaso);
ImporteTex        := FORMAT(VImporte);
NumTurnoTex       := FORMAT(NumTurno);
TipoContrapTex    := FORMAT(TipoContrapartida);

    IF ISCLEAR(XMLDom) THEN
      CREATE (XMLDom);

    IF ISCLEAR(SoapHttpConn) THEN
      CREATE(SoapHttpConn);
```

```
    SoapHttpConn.Property('EndPointURL', URLWS);
    SoapHttpConn.Connect;
    SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_RegDiario');
    SoapHttpConn.BeginMessage;
    IF ISCLEAR(SoapSerialize) THEN
      CREATE(SoapSerialize);
    SoapSerialize.Init(SoapHttpConn.InputStream);
    SoapSerialize.StartEnvelope('','STANDARD','');
    SoapSerialize.StartBody('');
    SoapSerialize.WriteXML('<TPV_RegDiario xmlns="http://tempuri.org/">'+
                    '<Usuario>'+Usuario+'</Usuario>'+
                    '<password>'+Password+'</password>'+
                    '<Fecharegistro>'+FechaText+'</Fecharegistro>'+
                    '<Tipodocumento>'+TipoDocTex+'</Tipodocumento>>'+
                    '<NDocumento>'+"NºDocumento"+'</NDocumento>'+
                    '<TipoMovimiento>'+TipoMovTex+'</TipoMovimiento>'+
                    '<NCuenta>'+"NºCuenta"+'</NCuenta>'+
                    '<VDescripcion>'+VDescripcion+'</VDescripcion>'+
                    '<VImporteTex>'+ImporteTex+'</VImporteTex>'+
'<TipoContrapartidaTex>'+TipoContrapTex+'</TipoContrapartidaTex>'+
                    '<NContrapartida>'+"NºContrapartida"+'</NContrapartida>'+
                    '<CodTienda>'+CodTienda+'</CodTienda>'+
                    '<CodTPV>'+CodTPV+'</CodTPV>'+
                    '<NTurnoTex>'+NumTurnoTex+'</NTurnoTex>'+
                    '<TipoMovTraspasoTex>'+TipoMovTraspTex+'</TipoMovTraspasoTex>'+
                    '<Signo>'+Signo +'</Signo>');

    SoapSerialize.WriteXML('</TPV_RegDiario>');
    SoapSerialize.EndBody;
    SoapSerialize.EndEnvelope;
    SoapHttpConn.EndMessage;

    IF SoapHttpConn.Error <> '' THEN
      EXIT(FALSE);

    IF ISCLEAR(XMLDom) THEN
      CREATE(XMLDom);

    XMLDom.async := FALSE;
    XMLDom.load(SoapHttpConn.OutputStream);
    XMLDom.save(cduILR.DameDirXML() + 'Diario.xml');

    IF ComprobarERROR(XMLDom,TxtError) THEN BEGIN
      EXIT(FALSE);
    END;
    EXIT(TRUE);
```

TPV_Traer_Pago_Central(Usuario,PassWord,NumDoc)

Esta función se invoca cuando el personal de tienda necesita insertar un bono en su base de datos, ya que un cliente quiere hacer el pago de su venta mediante bono o vale. Entonces, introducen el número de bono perteneciente al cliente y el sistema invoca

dicha función para comprobar si existe ese bono en la base de datos de CENTRAL. Si existe, el sistema realiza la inserción del bono. Por otro lado, se pueden dar dos errores: Que el bono esté consumido totalmente y el sistema impide utilizarlo de nuevo, o que el bono no exista en la base de datos de CENTRAL, porque no se haya registrado correctamente.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si EncabezadoWeb.Vacio entonces
    Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_Traer_Pago_Central);
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXml('<TPV_Traer_Pago_Central xmlns=http://tempuri.org/>'+
                '<Usuario>'+Usuario+'</Usuario>'+'<password>'+Password+
                '</password>'+'<NumDoc>'+NumDoc+'</NumDoc>'+
                '</TPV_Traer_Pago_Central>')
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>'' entonces
   Limpiar(EncabezadoWeb)
   ERROR('No se ha encontrado el pago en central. Intentelo de nuevo mas tarde')
Si EstaVacío(NodoXml) entonces
   Crear(NodoXml)
NodoXml.cargar(EncabezadoWeb)
NodoXml.guardar('PagoCentral.xml')
Si ComprobarError(NodoXml,txtError) entonces
   ERROR('%1',txtError)
NodoLista=NodoXml.CogerElementos('HistoricoAbonoVenta')
Si NodoLista.Longitud=0 entonces
   ERROR('No se ha encontrado el vale con numero %1. Revisa que lo has escrito
   Correctamente,NumDoc)
Sino
   Cuantos=NodoLista.Longitud
Si Cuantos>1 entonces
   ERROR('Se ha encontrado mas de un vale, afina la busqueda')
i=0
```

```
Repetir
   NodoLista=NodoXml.CogerElementos('HistoricoAbonoVenta')
   Si NodoLista.longitud>0 entonces
      mCod=NodoLista.elemento(i).texto
   Sino
      ERROR('No ha llegado el documento a CENTRAL')
   Limpiar(TipoDoc)
   NodoLista.CogerElementos('ImportePendiente')
   Si NodoLista.longitud>0 entonces
      TipoDoc=NodoLista.elemento(i).texto
      Si TipoDoc<>verdadero entonces
         Msg=TPV_Traer_Consumos_Vale(Usuario,Password,NumDoc)
         ERROR(Msg)
   Si NO TablaBonos.Coger(mCod) entonces
       TablaBonos se inicializa
       TablaBonos.NoDocReg=mCod
       Insertar registro TablaBonos
   NodoLista=NodoXml.CogerElementos(NombreEmpresa)
   Si NodoLista.Longitud>0 entonces
      TablaBonos.NombreEmpresa=NodoLista.elemento(i).texto
   NodoLista=NodoXml.CogerElementos(CodClienteOrigen)
   Si NodoLista.longitud>0 entonces
      TablaBonos.CodClienteOrigen=NodoLista.elemento(i).texto
      CodCliente=TablaBonos.CodClienteOrigen
   NodoLista.CogerElementos(ImporteInicial)
   Si NodoLista.Longitud>0 entonces
       ImporteTex=FormatearImporteDEC(NodoLista.elemento(i).texto)
       Evaluar(ImportePdte,ImporteTex)
       TablaBonos.ImporteInicial=ImportePdte
    Limpiar(TipoDoc)
    NodoLista=NodoXml.CogerElementos(TipoDocumento)
    Si NodoLista.longitud>0 entonces
       Caso TipoDoc
          '': TablaBonos.TipoDocumento=''
         'Bono':TablaBonos.TipoDocumento=Bono
         'Abono':TablaBonos.TipoDocumento=Abono
         'Reserva':TablaBonos.TipoDocumento=Reserva
    NodoLista=NodoXml.CogerElementos(ImporteReserva)
    Si NodoLista.longitud>0 entonces
       ImporteTex=FormatearImporteDEC(NodoLista.elemento(i).texto)
       Evaluar(ImportePdte,ImporteTex)
       TablaBonos.ImporteReserva=ImportePdte
    NodoLista=NodoXml.CogerElementos(ImporteBono)
    Si NodoLista.longitud>0 entonces
       ImporteTex=FormatearImporteDEC(NodoLista.elemento(i).texto)
       Evaluar(ImportePdte,ImporteTex)
       TablaBonos.ImporteBono=ImportePdte
    NodoLista=NodoXml.CogerElementos(ImporteAbono)
    Si NodoLista.longitud>0 entonces
       ImporteTex=FormatearImporteDEC(NodoLista.elemento(i).texto)
       Evaluar(ImportePdte,ImporteTex)
       TablaBonos.ImporteAbono=ImportePdte
   Limpiar(TipoDoc)
   NodoLista.CogerElementos('Pendiente')
```

```
    Si NodoLista.longitud>0 entonces
       TipoDoc=NodoLista.elemento(i).texto
       Si TipoDoc=verdadero entonces
          TablaBonos.Pendiente=Verdadero
       Sino
          TablaBonos.Pendiente=Falso
     NodoLista=NodoXml.CogerElementos(FechaUltModificacion)
     Si NodoLista.longitud>0 entonces
        Evaluar(FechaRegistro,CopiaCadena(NodoLista.elemento(i).texto,9,2)+'/'+
        CopiaCadena(NodoLista.elemento(i).texto,6,2)+'/'+CopiaCadena(
        NodoLista.elemento(i).texto,1,4))
        TablaBonos.FechaUltModificacion=FechaRegistro
     NodoLista=NodoXml.CogerElementos(Importeliquidado)
     Si NodoLista.longitud>0 entonces
         ImporteTex=FormatearImporteDEC(NodoLista.elemento(i).texto)
         Evaluar(ImportePdte,ImporteTex)
         TablaBonos.ImporteLiquidado=ImportePdte
    NodoLista=NodoXml.CogerElementos(FechaCreacion)
    Si NodoLista.longitud>0 entonces
        Evaluar(FechaRegistro,CopiaCadena(NodoLista.elemento(i).texto,9,2)+'/'+
        CopiaCadena(NodoLista.elemento(i).texto,6,2)+'/'+CopiaCadena(
        NodoLista.elemento(i).texto,1,4))
        TablaBonos.FechaCreacion=FechaRegistro
    Modificar registro TablaBonos
    i=i+1
Hasta i=Cuantos
Exit(Cuantos)
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF Conf.FINDFIRST() THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
  END
ELSE
  ERROR('No Existe Configuración TPV')
BEGIN
 IF ISCLEAR(SoapHttpConn) THEN
   CREATE(SoapHttpConn);

 SoapHttpConn.Property('EndPointURL', URLWS);
 SoapHttpConn.Connect;
 SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_Traer_Pago_Central');
 SoapHttpConn.BeginMessage;
 IF ISCLEAR(SoapSerialize) THEN
   CREATE(SoapSerialize);
 SoapSerialize.Init(SoapHttpConn.InputStream);
 SoapSerialize.StartEnvelope('','STANDARD','');
 SoapSerialize.StartBody('');
 SoapSerialize.WriteXML('<TPV_Traer_Pago_Central xmlns="http://tempuri.org/">'+
```

```
                '<Usuario>'+Usuario+'</Usuario>'+
                '<password>'+Password+'</password>'+
                '<NumDoc>'+NumDoc +'</NumDoc>'+
                '</TPV_Traer_Pago_Central>');

  SoapSerialize.EndBody;
  SoapSerialize.EndEnvelope;
  SoapHttpConn.EndMessage;
  IF  SoapHttpConn.Error <> '' THEN BEGIN
     CLEAR(SoapHttpConn);
     ERROR(Text0010);
  END;

  IF ISCLEAR(XMLDom) THEN
    CREATE(XMLDom);
  XMLDom.async := FALSE;
  XMLDom.load(SoapHttpConn.OutputStream);
  XMLDom.save(cduILR.DameDirXML() + 'PagoCentral.xml');

  IF ComprobarERROR(XMLDom,TxtError) THEN
   ERROR('%1',TxtError);

  DOMNodelist:=
XMLDom.getElementsByTagName('HistabonoventaMultiempNdocumentoregistrado');
  IF DOMNodelist.length = 0 THEN
    ERROR(Text0009,NumDoc)
  ELSE
    Cuantos := DOMNodelist.length;

IF Cuantos > 1 THEN
    ERROR('Se ha encontrado más de 1 vale con el número %1. Afina la búsqueda',NumDoc);
  i:=0;
  REPEAT
    DOMNodelist:=
XMLDom.getElementsByTagName('HistabonoventaMultiempNdocumentoregistrado');
    IF DOMNodelist.length > 0 THEN
      mCod := DOMNodelist.item(i).text
    ELSE
      ERROR('No ha llegado el nº de documento al traer de central.');

   CLEAR(TipoDoc);
    DOMNodelist:= XMLDom.getElementsByTagName('HistabonoventaMultiempPendiente');
    IF DOMNodelist.length > 0 THEN BEGIN
      TipoDoc := DOMNodelist.item(i).text;
      IF TipoDoc <> 'true' THEN
      BEGIN
        Msg:= TPV_Traer_Consumos_Vale(Usuario,Password,NumDoc);
        ERROR(Msg);
      END;
    END;
    IF NOT Bonos.GET(mCod) THEN BEGIN
      Bonos.INIT;
      Bonos."Nº documento registrado" := mCod;
      Bonos.INSERT;
```

```
    END;
    DOMNodelist:=
XMLDom.getElementsByTagName('HistabonoventaMultiempNombreempresa');
    IF DOMNodelist.length > 0 THEN
      Bonos."Nombre empresa" := DOMNodelist.item(i).text;
    DOMNodelist:=
XMLDom.getElementsByTagName('HistabonoventaMultiempCodclienteorigen');
    IF DOMNodelist.length > 0 THEN BEGIN
      Bonos."Cod. cliente origen" := DOMNodelist.item(i).text;
      CodCliente := Bonos."Cod. cliente origen";
    END;
    DOMNodelist:=
XMLDom.getElementsByTagName('HistabonoventaMultiempImporteinicialDL');
    IF DOMNodelist.length > 0 THEN BEGIN
      ImporteTex := FormatearImporteDEC(DOMNodelist.item(i).text);
      EVALUATE(ImportePdte,ImporteTex);
      Bonos."Importe inicial (DL)" := ImportePdte;
    END;
    CLEAR(TipoDoc);
    DOMNodelist:=
XMLDom.getElementsByTagName('HistabonoventaMultiempTipodocumento');
    IF DOMNodelist.length > 0 THEN BEGIN
      TipoDoc := DOMNodelist.item(i).text;
      CASE TipoDoc OF
         ' ' : Bonos."Tipo documento" := Bonos."Tipo documento"::" ";
         'Bono' : Bonos."Tipo documento" := Bonos."Tipo documento"::Bono;
         'Abono' : Bonos."Tipo documento" := Bonos."Tipo documento"::Abono;
         'Reseva' : Bonos."Tipo documento" := Bonos."Tipo documento"::Reserva;
      END;
    END;
    DOMNodelist:=
XMLDom.getElementsByTagName('HistabonoventaMultiempImportereservaDL');
    IF DOMNodelist.length > 0 THEN BEGIN
      ImporteTex := FormatearImporteDEC(DOMNodelist.item(i).text);
      EVALUATE(ImportePdte,ImporteTex);
      Bonos."Importe reserva (DL)" := ImportePdte;
    END;
    DOMNodelist:=
XMLDom.getElementsByTagName('HistabonoventaMultiempImportebonoDL');
    IF DOMNodelist.length > 0 THEN BEGIN
      ImporteTex := FormatearImporteDEC(DOMNodelist.item(i).text);
      EVALUATE(ImportePdte,ImporteTex);
      Bonos."Importe bono (DL)" := ImportePdte;
    END;
    DOMNodelist:=
XMLDom.getElementsByTagName('HistabonoventaMultiempImporteabonoDL');
    IF DOMNodelist.length > 0 THEN BEGIN
      ImporteTex := FormatearImporteDEC(DOMNodelist.item(i).text);
      EVALUATE(ImportePdte,ImporteTex);
      Bonos."Importe abono (DL)" := ImportePdte;
    END;
    CLEAR(TipoDoc);
    DOMNodelist:= XMLDom.getElementsByTagName('HistabonoventaMultiempPendiente');
    IF DOMNodelist.length > 0 THEN BEGIN
```

```
      TipoDoc := DOMNodelist.item(i).text;
      IF TipoDoc = 'true' THEN
        Bonos.Pendiente := TRUE
      ELSE
        Bonos.Pendiente := FALSE;
    END;
    DOMNodelist:=
XMLDom.getElementsByTagName('HistabonoventaMultiempFechaultmodificación');
    IF DOMNodelist.length > 0 THEN BEGIN

EVALUATE(FechaRegistro,COPYSTR(DOMNodelist.item(i).text,9,2)+'/'+COPYSTR(DOMN
odelist.item(i).text,6,2)+'/'+
      COPYSTR(DOMNodelist.item(i).text,1,4));
      Bonos."Fecha ult. modificación" := FechaRegistro;
    END;
    DOMNodelist:=
XMLDom.getElementsByTagName('HistabonoventaMultiempImporteliquidadoDL');
    IF DOMNodelist.length > 0 THEN BEGIN
      ImporteTex := FormatearImporteDEC(DOMNodelist.item(i).text);
      EVALUATE(ImportePdte,ImporteTex);
      Bonos."Importe liquidado (DL)" := ImportePdte;
    END;
    DOMNodelist:=
XMLDom.getElementsByTagName('HistabonoventaMultiempFechacreación');
    IF DOMNodelist.length > 0 THEN BEGIN

EVALUATE(FechaRegistro,COPYSTR(DOMNodelist.item(i).text,9,2)+'/'+COPYSTR(DOMN
odelist.item(i).text,6,2)+'/'+
      COPYSTR(DOMNodelist.item(i).text,1,4));
      Bonos."Fecha creación" := FechaRegistro;
    END;
    Bonos.MODIFY;
    i:=i+1;
  UNTIL i= Cuantos;
  EXIT(Cuantos);
END;
```

TPV_Traer_Consumos_Vale(Usuario,Contraseña,NumDoc)

Esta función se invoca cuando un bono que se quiere traer de CENTRAL, el importe íntegro del mismo está consumido. Entonces, se llama a esta función para que muestre en qué ventas estaba inmerso el bono y así poder averiguar dónde puede estar el error.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
```

```
Si EncabezadoWeb.Vacio entonces
    Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_Traer_Consumos_Vale')
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXml('<TPV_Traer_Consumos_Vale xmlns=http://tempuri.org/>'
                +'<Usuario>'+Usuario+'</Usuario>'+'<password>'+Password+
                '</password>'+'<NumDoc>'+NumDoc+'</NumDoc>'+
                '</TPV_Traer_Consumos_Vale>')
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>'' entonces
   Limpiar(EncabezadoWeb);
   ERROR('ERROR de comunicaciones al traer consumos de vale de central. Inténtelo
   mas tarde)
Si EstaVacío(NodoXml) entonces
    Crear(NodoXml)
NodoXml.cargar(EncabezadoWeb)
NodoXml.guardar('ConsumosVale.xml')
Si ComprobarERROR(NodoXml,txtError) entonces
    ERROR('%1',txtError);
NodoLista=NodoXml.CogerElementos(MultiformadePago)
Si NodoLista.longitud=0 entonces
   Msg='---VALE'+Formato(NumDoc)+'CONSUMIDO---\\';
   Msg=Msg+'No se han encontrado detalle de los consumos para el vale con numero
   +Formato(NumDoc)
   Exit(Msg)
Cuantos=NodoLista.Longitud
Msg='---VALE+Formato(NumDoc)+'CONSUMIDO ---\\';
Msg=Msg+'CONSUMOS REALIZADOS\';
Msg=Msg+'-------------------------------------\';
i=0
Repetir
   NodoLista=NodoXml.cogerElementos(MultiformadePago)
   Msg=Msg+'\'+NodoLista.elemento(i).texto
   NodoLista=NodoXml.cogerElementos(FechaRegistro)
   Si NodoLista.longitud>0 entonces
      Evaluar(FechaRegistro,CopiaCadena(NodoLista.elemento(i).texto,9,2)+'/'+
      CopiaCadena(NodoLista.elemento(i).texto,6,2)+'/'+
      CopiaCadena(NodoLista.elemento(i).texto,1,4));
      Msg=Msg+'-'+Formato(FechaRegistro);
   NodoLista=NodoXml.CogerElementos(Importe)
   Si NodoLista.longitud>0 entonces
      ImporteTex=FormatearImporteDec(NodoLista.elemento(i).texto)
      Evaluar(ImportePdte,ImporteTex)
      Msg=Msg+'-'+Formatear(ImportePdte)
      i=i+1
```

Hasta i=Cuantos

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF Conf.FINDFIRST() THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
   END
ELSE
  ERROR('No Existe Configuración TPV');

BEGIN
 IF ISCLEAR(SoapHttpConn) THEN
    CREATE(SoapHttpConn);

 SoapHttpConn.Property('EndPointURL', URLWS);
 SoapHttpConn.Connect;
SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_Traer_Consumos_Vale');
 SoapHttpConn.BeginMessage;
 IF ISCLEAR(SoapSerialize) THEN
   CREATE(SoapSerialize);
 SoapSerialize.Init(SoapHttpConn.InputStream);
 SoapSerialize.StartEnvelope('','STANDARD','');
 SoapSerialize.StartBody('');

 SoapSerialize.WriteXML('<TPV_Traer_Consumos_Vale xmlns="http://tempuri.org/">'+
                  '<Usuario>'+Usuario+'</Usuario>'+
                  '<password>'+Password+'</password>'+
                  '<NumDoc>'+NumDoc +'</NumDoc>'+
                  '</TPV_Traer_Consumos_Vale>');

 SoapSerialize.EndBody;
 SoapSerialize.EndEnvelope;
 SoapHttpConn.EndMessage;
 IF  SoapHttpConn.Error <> '' THEN BEGIN
    CLEAR(SoapHttpConn);
    ERROR(Text0011);
 END;

 IF ISCLEAR(XMLDom) THEN
   CREATE(XMLDom);
 XMLDom.async := FALSE;
 XMLDom.load(SoapHttpConn.OutputStream);
 XMLDom.save(cduILR.DameDirXML() + 'ConsumosVale.xml')
 IF ComprobarERROR(XMLDom,TxtError) THEN
  ERROR('%1',TxtError);

 DOMNodelist:= XMLDom.getElementsByTagName('MultiformadePagoNDocumento');
 IF DOMNodelist.length = 0 THEN
```

```
  BEGIN
    Msg:='--- VALE '+FORMAT(NumDoc) + ' CONSUMIDO ---\\';
    Msg+='No se han encontrado detalle de los consumos para el vale con número
'+FORMAT(NumDoc);
    EXIT(Msg);
  END;
  Cuantos := DOMNodelist.length;
  Msg:='--- VALE '+FORMAT(NumDoc) + ' CONSUMIDO ---\\';
  Msg:=Msg + 'CONSUMOS REALIZADOS\';
  Msg:=Msg + '------------------------------\';
  i:=0;
  REPEAT
    DOMNodelist:= XMLDom.getElementsByTagName('MultiformadePagoNDocumento');
    Msg := Msg + '\'+DOMNodelist.item(i).text;
    DOMNodelist:= XMLDom.getElementsByTagName('MultiformadePagoFecharegistro');
    IF DOMNodelist.length > 0 THEN
    BEGIN
EVALUATE(FechaRegistro,COPYSTR(DOMNodelist.item(i).text,9,2)+'/'+COPYSTR(DOMN
odelist.item(i).text,6,2)+'/'+
      COPYSTR(DOMNodelist.item(i).text,1,4));
      Msg:=Msg + ' - ' +FORMAT(FechaRegistro);
    END;
    DOMNodelist:= XMLDom.getElementsByTagName('MultiformadePagoImporte');
    IF DOMNodelist.length > 0 THEN BEGIN
      ImporteTex := FormatearImporteDEC(DOMNodelist.item(i).text);
      EVALUATE(ImportePdte,ImporteTex);
      Msg:=Msg + ' - ' +cduILR.FormateaEuro(ImportePdte) ;
    END;
    i:=i+1;
  UNTIL i= Cuantos;
  EXIT(Msg);
END;
```

TPV_Cierre_Caja2(Usuario,Password,rcdCaja)

Esta función se invoca cuando el encargado de la tienda hace el cierre del dia. Entonces, esta función manda todos los datos correspondientes a la caja del día de trabajo, es decir, ventas efectivo, ventas tarjeta, ventas con bono, abonos efectivo, etc. Así, el personal de contabilidad tiene todos los datos correspondientes con las cajas de las tiendas, para poder ver si tienen descuadre o por si quieren ver alguna venta en particular.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTienda sea rcdCaja.CodTienda
ConfigTpv se filtra CodTpv sea rcdCaja.CodTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
```

```
Sino
   ERROR('No existe Configuración TPV');
TablaCaja se inicializa
TablaCaja se filtra FechaInicioDia sea rcdCaja.FechaInicioDia
nvNumCajas=0
cvCodTpv=''
Si encuentra registros TablaCaja entonces
    Repetir
      Si TablaCaja.CodTpv<>cvCodTpv
          nvNumCajas=nvNumCajas+1
          cvCodTpv=TablaCaja.CodTpv
    Hasta siguiente registro TablaCaja sea vacío
FechaInicio=Formato(rcdCaja.FechaInicioDia)
HoraInicio=Formato(rcdCaja.HoraInicioDia)
FechaFin=Formato(rcdCaja.FechafinDia)
HoraFin=Formato(rcdCaja.HoraFinDia)
SaldoIni=Formato(rcdCaja.SaldoInicial)
Descuadre=Formato(rcdCaja.DiferenciaCaja)
Recuento=Formato(rcdCaja.Totalrecuento)
CambioFinal=Formato(rcdCaja.SaldoFinalTpv)
Descuadre_Tarjeta=Formato(rcdCaja.DiferenciaTarjeta)
Recuento_Datafono=Formato(rcdCaja.TotalrecuentoTarjeta)
rcdCaja.CalculaCampo(TotalgastosTpv,VentasTarjeta,VentasEfectivo,VentasBonoPoly,
          Reservastarjeta,Reservasefectivo,Reservas BonoPoly,Abonostarjeta,
          AbonosEfectivo,AbonosBonoPoly,ReservaCanceladatarjeta,
         Reservacanceladaefectivo,ReservaCanceladaBono)
Si EstaVacío(NodoXml) entonces
    Crear(NodoXml)
Si EncabezadoWeb.Vacio entonces
    Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_CierreCaj2_Enviar');
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXml('<TPV_CierreCaja2_Enviar xmlns=http://tempuri.org/>'+
                '<Usuario>'+Usuario+'</Usuario>'+'<password>'+Password+
                '</password>'+'<CodTienda>'+rcdCaja.CodTienda+'</CodTienda>'+
                '<CodTpv>'+rcdCaja.CodTpv+'</CodTpv>'+'<FechaInicio>'+
                FechaInicio+'</FechaInicio>'+'<HoraInicio>'+HoraInicio+
                '</HoraInicio>'+'<FechaFin>'+FechaFin+'</FechaFin>'+'<HoraFin>'+
                HoraFin+'</HoraFin>'+'<CambioInicial>'+SaldoIni+'</CambioInicial>
                +'<Descuadre>'+Descuadre+'</Descuadre>'+'<Recuento>'+Recuento+
                '</Recuento>'+<CambioFinal>+CambioFinal+'</CambioFinal>'+
                '<Descuadre_Tarjeta>'+Descuadre_Tarjeta+'</Descuadre_Tarjeta>'+
                '<Recuento_Datafono>'+Recuento_Datafono+'</Recuento_Datafono>'
                +'<Numero_Cajas>'+Formateo(nvNumCajas)+'</Numero_Cajas>'+
                +'<Gastos>+Formateo(rcdCaja.TotalGastosTpv)+'</Gastos>'+
                '<Ventas_Tarjeta>'+Formateo(rcdCaja.VentasTarjeta)+
                '</Ventas_Tarjeta>'+'<Ventas_Efectivo>'+
```

```
                Formateo(rcdCaja.VentasEfectivo)+'</Ventas_Efectivo>'+
               '<Ventas_Bono>'+Formateo(rcdcaja.VentasBono)+'</Ventas_Bono>'
              +'<Reservas_Tarjeta>'+Formateo(rcdCajas-ReservasTarjeta)+
               '</Reservas_Tarjeta>'+'<Reservas_Efectivo>'+
              Formateo(rcdCaja.ReservasEfectivo)+'</Reservas_Efectivo>'+
            '<Reservas_Bono>'+Formateo(rcdCaja.ReservasBono)+
            '</Reservas_Bono>'+'<Abonos_Tarjeta>'+
            Formateo(rcdCaja.AbonosTarjeta)+'</Abonos_Tarjeta>'+
            '<Abonos_Efectivo>'+Formateo(rcdCaja.AbonosEfectivo)+
            '</Abonos_Efectivo>'+'<Abonos_Bono>'+
            Formateo(rcdCaja.AbonosBono)+'</Abonos_Bono>'+
            '<Reserva_Cancel_Tarjeta>'+Formateo(rcdCaja.ReservasCancelTarjeta)+
            '</Reserva_Cancel_Tarjeta>'+'<Reserva_Cancel_Efectivo>'+
            Formateo(rcdCaja.ReservaCancelEfectivo)+'</Reserva_Cancel_Efectivo>'
            +'<Reserva_Cancel_Bono>'+Formateo(rcdCaja.ReservaCancelBono)+
            +'</Reserva_Cancel_Bono>'
ConectorWeb.EscribirXML('</TPV_CierreCaja2_Enviar>');
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>'' entonces
   Salir(Falso)
Si EstaVacío(NodoXml) entonces
    Crear(NodoXml)
NodoXml.Cargar(EncabezadoWeb)
NodoXml.Guardar('CierreCaja2.xml');
Si ComprobarError(NodoXml,txtError) entonces
    Salir(Falso)
Salir(Verdadero)
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. tienda", rcdCaja."Cód. tienda");
Conf.SETRANGE("Cód. TPV", rcdCaja."Cód. TPV");
IF Conf.FINDFIRST() THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
   END
ELSE
  ERROR('No Existe Configuración TPV');

rcdCajaAux.RESET;
rcdCajaAux.SETRANGE("Fecha inicio día",rcdCaja."Fecha inicio día");
nvNumCajas:=0;
cvCodTpvAux:='';
IF rcdCajaAux.FINDSET THEN
BEGIN
 REPEAT
    IF rcdCajaAux."Cód. TPV" <> cvCodTpvAux THEN
    BEGIN
      nvNumCajas+=1;
```

```
      cvCodTpvAux:=rcdCajaAux."Cód. TPV";
    END;
  UNTIL rcdCajaAux.NEXT=0;
END;

FechaInicio   := FORMAT(rcdCaja."Fecha inicio día");
HoraInicio    := FORMAT(rcdCaja."Hora inicio día");
FechaFin      := FORMAT(rcdCaja."Fecha fin día");
HoraFin       := FORMAT(rcdCaja."Hora fin día");
SaldoIni      := FORMAT(rcdCaja."Saldo inicial");
Descuadre     := FORMAT(rcdCaja."Diferencia caja");
Recuento      := FORMAT(rcdCaja."Total recuento");
CambioFinal   := FORMAT(rcdCaja."Saldo final TPV");
Descuadre_Tarjeta  := FORMAT(rcdCaja."Diferencia tarjeta");
Recuento_Datafono  := FORMAT(rcdCaja."Total recuento Tarjeta");


rcdCaja.CALCFIELDS("Total gastos tpv","Ventas Tarjeta","Ventas Efectivo","Ventas
BonoPoly",
              "Reservas Tarjeta","Reservas Efectivo","Reservas BonoPoly",
              "Abonos Tarjeta","Abonos Efectivo","Abonos BonoPoly",
              "Reservas Cancelada Tarjeta","Reservas Cancelada Efectivo","Reservas Cancelada
Bono");

 BEGIN
   window.OPEN(Text000 + '@1@@@@@@@@@@@@@@@@@@@@@\');
   window.UPDATE(1,0);

   IF ISCLEAR(XMLDom) THEN
     CREATE (XMLDom);

   IF ISCLEAR(SoapHttpConn) THEN
     CREATE(SoapHttpConn);

   SoapHttpConn.Property('EndPointURL', URLWS);
   SoapHttpConn.Connect;

   SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_CierreCaja2_Enviar');
   SoapHttpConn.BeginMessage;
   IF ISCLEAR(SoapSerialize) THEN
     CREATE(SoapSerialize);
   SoapSerialize.Init(SoapHttpConn.InputStream);
   SoapSerialize.StartEnvelope('','STANDARD','');
   SoapSerialize.StartBody('');
   SoapSerialize.WriteXML('<TPV_CierreCaja2_Enviar xmlns="http://tempuri.org/">'+
                    '<Usuario>'+Usuario+'</Usuario>'+
                    '<password>'+Password+'</password>'+
                    '<CodTienda>'+rcdCaja."Cód. tienda"+'</CodTienda>'+
                    '<CodTPV>' + rcdCaja."Cód. TPV" + '</CodTPV>' +
                    '<FechaInicio>' + FechaInicio + '</FechaInicio>' +
                    '<HoraInicio>' + HoraInicio + '</HoraInicio>' +
                    '<FechaFin>' + FechaFin + '</FechaFin>' +
                    '<HoraFin>' + HoraFin + '</HoraFin>' +
                    '<CambioInicial>' + SaldoIni + '</CambioInicial>' +
```

```
                    '<Descuadre>' + Descuadre + '</Descuadre>' +
                    '<Recuento>' + Recuento + '</Recuento>' +
                    '<CambioFinal>' + CambioFinal + '</CambioFinal>' +
                    '<Descuadre_Tarjeta>' + Descuadre_Tarjeta + '</Descuadre_Tarjeta>' +
                    '<Recuento_Datafono>' + Recuento_Datafono + '</Recuento_Datafono>' +

                    '<Numero_Cajas>' + FORMAT(nvNumCajas) + '</Numero_Cajas>' +
                    '<Gastos>' + FORMAT(rcdCaja."Total gastos tpv") + '</Gastos>' +
                    '<Ventas_Tarjeta>' + FORMAT(rcdCaja."Ventas Tarjeta") +
'</Ventas_Tarjeta>' +
                    '<Ventas_Efectivo>' + FORMAT(rcdCaja."Ventas Efectivo") +
'</Ventas_Efectivo>' +
                    '<Ventas_Bono>' + FORMAT(rcdCaja."Ventas BonoPoly") +
'</Ventas_Bono>' +
                    '<Reservas_Tarjeta>' + FORMAT(rcdCaja."Reservas Tarjeta") +
'</Reservas_Tarjeta>' +
                    '<Reservas_Efectivo>' + FORMAT(rcdCaja."Reservas Efectivo") +
'</Reservas_Efectivo>' +
                    '<Reservas_Bono>' + FORMAT(rcdCaja."Reservas BonoPoly") +
'</Reservas_Bono>' +
                    '<Abonos_Tarjeta>' + FORMAT(rcdCaja."Abonos Tarjeta") +
'</Abonos_Tarjeta>' +
                    '<Abonos_Efectivo>' + FORMAT(rcdCaja."Abonos Efectivo") +
'</Abonos_Efectivo>' +
                    '<Abonos_Bono>' + FORMAT(rcdCaja."Abonos BonoPoly") +
'</Abonos_Bono>' +
                  '<Reservas_Cancel_Tarjeta>' + FORMAT(rcdCaja."Reservas Cancelada
Tarjeta") + '</Reservas_Cancel_Tarjeta>' +
                '<Reservas_Cancel_Efectivo>' + FORMAT(rcdCaja."Reservas Cancelada
Efectivo") + '</Reservas_Cancel_Efectivo>' +
                    '<Reservas_Cancel_Bono>' + FORMAT(rcdCaja."Reservas Cancelada Bono")
+ '</Reservas_Cancel_Bono>');



    SoapSerialize.WriteXML('</TPV_CierreCaja2_Enviar>');
    SoapSerialize.EndBody;
    SoapSerialize.EndEnvelope;
    SoapHttpConn.EndMessage;
    window.CLOSE;

    IF SoapHttpConn.Error <> '' THEN
      EXIT(FALSE);

    IF ISCLEAR(XMLDom) THEN
      CREATE(XMLDom);

    XMLDom.async := FALSE;
    XMLDom.load(SoapHttpConn.OutputStream);


    XMLDom.save(cduILR.DameDirXML() + 'CierreCaja2.xml');

    IF ComprobarERROR(XMLDom,TxtError) THEN BEGIN
```

```
    EXIT(FALSE);
  END;
  EXIT(TRUE);
```

TPV_AltaManualBono(Usuario,VBono)

Esta función se invoca cuando alguna de las promociones vigentes en la tienda genera un vale descuento. Entonces, una vez creado de forma local en la base de datos, se debe centralizar dicho bono mediante esta función, para que a la hora de consumirlo no haya ningún problema en su utilización.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si EncabezadoWeb.Vacio entonces
    Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_AltaManualBono);
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
tmpEmp:=VBono.NombreEmpresa
Si PosicionCadena(VBono.NombreEmpresa,'&')<>0 entonces
   tmpEmp=CopiaCadena(VBono.NombreEmpresa,1,
   PosicionCadena(VBono.NombreEmpresa,'&')-1);
   tmpEmp=tmpEmp+'&amp;'+CopiaCadena(VBono.NombreEmpresa,
   PosicionCadena(VBono.NombreEmpresa,'&')+1)
ConectorWeb.EscribirXml('<TPV_AltaManualBono xmlns=http://tempuri.org/>'+
                          '<Usuario>'+Usuario+'</Usuario>'+'<Password>'+
                        Password+'</password>'+'<numero>+
                        VBono.NoDocRegistrado+'</numero>'+'<Empresa>'+
                        tmpEmp+'</Empresa>'+'<Cliente>'+VBono.Cliente+
                        '</Cliente>'+'<Importe>'+Formateo(VBono.ImporteInicial)+
                        '</Importe>'+'<FechaEmision>'+
                        Formateo(VBono.FechaCreacion)+'</FechaEmision>'+
                        '<FechaCaducidad>'+Formateo(VBono.FechaCaducidad)+
                        '</FechaCaducidad>'+'</TPV_AltaManualBono>');
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>'' entonces
```

```
    TextoError=EncabezadoWeb.Error
    Limpiar(EncabezadoWeb)
    Salir(FALSO)
Sino
    VBono.ErrorEnvio=FALSO
    Modificar VBono
    Limpiar(EncabezadoWeb)
    Salir(VERDADERO)
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF Conf.FINDFIRST() THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
  END
ELSE
  ERROR('No Existe Configuración TPV');



BEGIN
 IF ISCLEAR(SoapHttpConn) THEN
   CREATE(SoapHttpConn);
 SoapHttpConn.Property('EndPointURL', URLWS);
 SoapHttpConn.Connect;
 SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_AltaManualBono');
 SoapHttpConn.BeginMessage;
 IF ISCLEAR(SoapSerialize) THEN
   CREATE(SoapSerialize);
 SoapSerialize.Init(SoapHttpConn.InputStream);
 SoapSerialize.StartEnvelope('','STANDARD','');
 SoapSerialize.StartBody('');

 tmpEmp:=VBono."Nombre empresa";
 IF STRPOS(VBono."Nombre empresa",'&') <> 0 THEN
 BEGIN
   tmpEmp:=COPYSTR(VBono."Nombre empresa",1,STRPOS(VBono."Nombre
empresa",'&')-1);
   tmpEmp:=tmpEmp + '&amp;' + COPYSTR(VBono."Nombre
empresa",STRPOS(VBono."Nombre empresa",'&')+1);
 END;
 SoapSerialize.WriteXML('<TPV_AltaManualBono xmlns="http://tempuri.org/">'+
                  '<Usuario>'+Usuario+'</Usuario>'+
                  '<password>'+Password+'</password>'+
                  '<numero>'+VBono."Nº documento registrado" +'</numero>'+
                  '<Empresa>'+ tmpEmp +'</Empresa>'+
                  '<Cliente>' + VBono."Cod. cliente origen" + '</Cliente>' +
                  '<Importe>' + FORMAT(VBono."Importe inicial (DL)") + '</Importe>' +
                  '<FechaEmision>' + FORMAT(VBono."Fecha creación") + '</FechaEmision>'
+
```

```
                '<FechaCaducidad>'          +FORMAT(VBono."Fecha          Caducidad")+
'</FechaCaducidad>'+
                '</TPV_AltaManualBono>');

  SoapSerialize.EndBody;
  SoapSerialize.EndEnvelope;
  SoapHttpConn.EndMessage;
  IF  SoapHttpConn.Error <> '' THEN
     BEGIN
       TextoError:=SoapHttpConn.Error;
       CLEAR(SoapHttpConn);
       CLEAR(SoapHttpConn);
       EXIT(FALSE)
     END
  ELSE
     BEGIN
       VBono."Error Envio":=FALSE;
       VBono.MODIFY;
       CLEAR(SoapHttpConn);
       EXIT(TRUE);
     END;
END;
```

TPV_Sincronizar_Producto(ParProducto)

Esta función se invoca cuando se quiere actualizar un producto al momento, debido a que se han realizado cambios recientemente en cuanto a precio, descripción, subfamilia, etc. Dicha funcionalidad se encuentra en la ficha de producto. Además, esta función se invoca para hacer la actualización diaria, ya que se ejecuta para cada producto modificado los días anteriores a la apertura.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
       UrlWs:= ConfigTpv.URLCentral
   Sino
       ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si EncabezadoWeb.Vacio entonces
    Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_Sincronizar_Producto');
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
```

```
ConectorWeb.EscribirXML('<TPV_Sincronizar_Producto xmlns="http//tempuri.org">'
                        +'<Usuario>'+ConfigTpv.CodigoTienda+'</Usuario>'+
                        '<password>'+ConfigTpv.password+'</password>'+
                        '<producto>'+ParProducto+'</producto>'+
                        '<TPV_Sincronizar_producto>');
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>'' entonces
   Limpiar(EncabezadoWeb)
   ERROR('Error al actualizar el producto %1, ParProducto)
Si EstaVacio(NodoXml) entonces
   Crear(NodoXml)
NodoXml.Cargar(EncabezadoWeb)
NodoXml.Guardar('Producto.xml');
NodoLista=NodoXML.cogerelementos('NoProducto')
Si Longitud(NodoLista)=0 entonces
    Salir(0)
Sino
   Cuantos=NodoLista.Longitud
NodoLista=NodoXml.CogerElementos('CodigoBarrasAsociado')
Si Longitud(NodoLista)=0 entonces
    Salir(0)
Sino
    CuantosBar=NodoLista.Longitud
i=0
Repetir
    NodoLista=NodoXML.cogerelementos('NoProducto')
    mCod=NodoLista.elemento(i).texto
    Si CodAnterior<>mCod entonces
        TablaProducto se inicializa
        TablaProducto se filtra No sea mCod
        Si no encuentra registros TablaProducto entonces
           TablaProducto se inicializa
           TablaProducto.No=NodoLista.elemento(i).texto
           Insertar registro TablaProducto
        Si Idioma<>1034 entonces
          NodoLista=NodoXMl.CogerElementos('Dimension1Code')
          TablaProducto.Class=NodoLista.elemento(i).texto
          NodoLista=NodoXMl.CogerElementos('Description')
          TablaProducto.Description2=CopiaCadena(NodoLista.elemento(i).texto,1,50)
          NodoLista=NodoXMl.CogerElementos('Description2')
          Si NodoLista.elemento(i).texto<>'' entonces
               TablaProducto.Description=CopiaCadena(NodoLista.elemento(i).texto,1,50)
          Sino
              TablaClase se inicializa
              TablaClase se filtra Codigo sea TablaProducto.Class
              Si encuentra registro TablaClase entonces
                 TablaProducto.Description=TablaClase.Description
        Sino
           NodoLista=NodoXMl.CogerElementos('Description')
           TablaProducto.Description=CopiaCadena(NodoLista.elemento(i).texto,1,50)
      NodoLista=NodoXMl.CogerElementos('UnidadMedida')
      TablaUnidadMedida se inicializa
```

```
      TablaUnidadMedida se filtra NoProducto sea mCod
      Si no encuentra registros TablaUnidadMedida entonces
         TablaUnidadMedida se inicializa
         TablaUnidadMedida.NoProducto=mCod
         TablaUnidadMedida.Code=NodoLista.elemento(i).texto
         TablaUnidadMedida.CantidadPorUnidadMedida=1
          Inserta registro TablaUnidadMedida
      TablaProducto.UnidadMedida=NodoLista.elemento(i).texto
      NodoLista=NodoXML.CogerElementos('GrupoRegistro')
      TablaProducto.GrupoRegistro=NodoLista.elemento(i).texto
      NodoLista=NodoXML.CogerElementos('GrupoDescuento')
      TablaProducto.GrupoDescuento=NodoLista.elemento(i).texto
      Si Idioma<>1034 entonces
         NodLista=NodoXml.CogerElemento(PrecioPsicoPortugal)
      Sino
        NodoLista=NodoXml.CogerElemento(PrecioPsico)
      Si NodoLista.elemento(i).texto<>'' entonces
         Precio=FormatearImporte(NodoLista.elemento(i).texto)
         Evaluar(PrecioDec,Precio)
         TablaProducto.Precio=Redóndear(PrecioDec,0.01)
      Producto.IncluyeIVA=Verdadero
      NodoLista=NodoXML.CogerElementos('NoProveedor')
      TablaProducto. NoProveedor =NodoLista.elemento(i).texto
      NodoLista=NodoXML.CogerElementos('CodigoProveedor')
      TablaProducto. CodigoProveedor =NodoLista.elemento(i).texto
      NodoLista=NodoXML.CogerElementos('GrupoNegocio')
      TablaProducto.GrupoNegocio=NodoLista.elemento(i).texto
      NodoLista=NodoXML.CogerElementos('GrupoIVA')
      TablaProducto.GrupoIVA=NodoLista.elemento(i).texto
      NodoLista=NodoXml.CogerElementos(FechaUltimaModTPV)
      Evaluar(FechaRegistro, CopiaCadena(NodoLista.elemento(i).texto,9,2)+
        '/'+CopiaCadena(NodoLista.elemento(i).texto,6,2)+'/'+
        CopiaCadena(NodoLista.elemento(i).texto,1,4));
        TablaProducto.FechaUltMod=FechaRegistro
        Modificar registro TablaProducto
  i=i+1
 CodAnterior=mCod
Hasta i=Cuantos
i=0
CodAnterior=''
Repetir
   NodoLista=NodoXml.CogerElementos(CodigoBarrasAlternativo)
   mCod=NodoLista.elemento(i).texto
   Si CodAnterior<>mCod entonces
      NodoLista=NodoXml.CogerElementos(CodigoBarrasAsociado)
      Si NodoLista.Longitud>0 entonces
         CodBarra=NodoLista.elemento(i).texto
         TablaCodBarra se inicializa
         TablaCodBarra se filtra Producto sea mCod
         Borrar todos los registros TablaCodBarra
         TablaCodBarra se inicializa
         TablaCodBarra.Producto=mCod
         TablaCodBarra.CodBarrasAsociado=CodBarra
         Insertar registro TablaCodBarra
```

```
   Sino
     NodoLista=NodoXml.CogerElementos(CodigoBarrasAsociado)
     Si NodoLista.Longitud>0 entonces
        TablaCodBarra se inicializa
        TablaCodBarra.Producto=mCod
        TablaCodBarra.CodBarrasAsociado=CodBarra
        Insertar registro TablaCodBarra
  i=i+1
  CodAnterior=mCod
Hasta i=CuantosBar
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF Conf.FINDFIRST() THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
  END
ELSE
  ERROR('No Existe Configuración TPV');

BEGIN
 IF ISCLEAR(SoapHttpConnLocal) THEN
   CREATE(SoapHttpConnLocal);


 SoapHttpConnLocal.Property('EndPointURL', URLWS);
 SoapHttpConnLocal.Connect;
SoapHttpConnLocal.Property('SoapAction','http://tempuri.org/TPV_Sincronizar_Producto');
 SoapHttpConnLocal.BeginMessage;
 IF ISCLEAR(SoapSerializeLocal) THEN
   CREATE(SoapSerializeLocal);
 SoapSerializeLocal.Init(SoapHttpConnLocal.InputStream);
 SoapSerializeLocal.StartEnvelope('','STANDARD','');
 SoapSerializeLocal.StartBody('');

 SoapSerializeLocal.WriteXML('<TPV_Sincronizar_Producto xmlns="http://tempuri.org/">'+
                  '<Usuario>'+Conf."Cód. tienda"+'</Usuario>'+
                  '<password>'+Conf.Password+'</password>'+
                  '<Producto>'+PARProducto +'</Producto>'+
                   '</TPV_Sincronizar_Producto>');

 SoapSerializeLocal.EndBody;
 SoapSerializeLocal.EndEnvelope;
 SoapHttpConnLocal.EndMessage;
 IF  SoapHttpConnLocal.Error <> '' THEN BEGIN
    CLEAR(SoapHttpConnLocal);
    ERROR(Text0012,PARProducto);
 END;

 IF ISCLEAR(XMLDomLocal) THEN
```

```
    CREATE(XMLDomLocal);
  XMLDomLocal.async := FALSE;
  XMLDomLocal.load(SoapHttpConnLocal.OutputStream);
  XMLDomLocal.save(cduILR.DameDirXML() + 'Producto.xml');

  DOMNodelistLocal:= XMLDomLocal.getElementsByTagName('ItemNo');
  IF DOMNodelistLocal.length = 0 THEN
    EXIT(0)
  ELSE
    Cuantos := DOMNodelistLocal.length;

  DOMNodelistLocal:=
XMLDomLocal.getElementsByTagName('CodigosbarrasalternativosCodbarrasasociado');
  IF DOMNodelistLocal.length = 0 THEN
    EXIT(0)
  ELSE
    CuantosBar := DOMNodelistLocal.length;

  i:=0;
  CodAnterior := '';
  REPEAT
    DOMNodelistLocal:= XMLDomLocal.getElementsByTagName('ItemNo');
    mCod:=DOMNodelistLocal.item(i).text;

    IF CodAnterior <> mCod THEN
    BEGIN
      Producto.RESET;
      Producto.SETRANGE(Producto."No.",mCod);
      IF NOT Producto.FINDFIRST() THEN
      BEGIN
        Producto.INIT;
        DOMNodelistLocal:= XMLDomLocal.getElementsByTagName('ItemNo');
        Producto."No.":=DOMNodelistLocal.item(i).text;
        Producto.INSERT;
      END;
      DOMNodelistLocal:=
XMLDomLocal.getElementsByTagName('ItemGlobalDimension1Code');
      Producto.Class:=DOMNodelistLocal.item(i).text;

      IF cduILR.ActivarIdioma() <> 1034 THEN
      BEGIN
        DOMNodelistLocal:= XMLDomLocal.getElementsByTagName('ItemDescription');
        Producto."Description 2":=COPYSTR(DOMNodelistLocal.item(i).text,1,50);
        DOMNodelistLocal:= XMLDomLocal.getElementsByTagName('ItemDescription2');
        IF DOMNodelistLocal.item(i).text <> '' THEN
        BEGIN
          Producto.Description:=COPYSTR(DOMNodelistLocal.item(i).text,1,50);
        END
        ELSE
        BEGIN
          rcdClase.RESET;
          rcdClase.SETRANGE(rcdClase.Code,Producto.Class);
          IF rcdClase.FINDFIRST() THEN
            Producto.Description:=rcdClase.Description;
```

```
      END;
    END
    ELSE
    BEGIN
      DOMNodelistLocal:= XMLDomLocal.getElementsByTagName('ItemDescription');
      Producto.Description:=COPYSTR(DOMNodelistLocal.item(i).text,1,50);
    END;
    DOMNodelistLocal:=
XMLDomLocal.getElementsByTagName('ItemBaseUnitofMeasure');
    UDProducto.RESET;
    UDProducto.SETRANGE(UDProducto."Item No.",mCod);
    IF NOT UDProducto.FINDFIRST() THEN
    BEGIN
      UDProducto.INIT;
      UDProducto."Item No.":=mCod;
      UDProducto.Code:=DOMNodelistLocal.item(i).text;
      UDProducto."Qty. per Unit of Measure":=1;
      UDProducto.INSERT;
    END;
    Producto."Base Unit of Measure" :=DOMNodelistLocal.item(i).text;
    DOMNodelistLocal:=
XMLDomLocal.getElementsByTagName('ItemInventoryPostingGroup');
    Producto."Inventory Posting Group":=DOMNodelistLocal.item(i).text;
    DOMNodelistLocal:= XMLDomLocal.getElementsByTagName('ItemItemDiscGroup');
    Producto."Item Disc. Group":=DOMNodelistLocal.item(i).text;



   IF cduILR.ActivarIdioma() <> 1034 THEN
      DOMNodelistLocal:=
XMLDomLocal.getElementsByTagName('ItemPrecioPsicoPortugal')
    ELSE
      DOMNodelistLocal:= XMLDomLocal.getElementsByTagName('ItemPrecioPsico');

    IF DOMNodelistLocal.item(i).text <> '' THEN
    BEGIN
      Precio:=FormatearImporteDEC(DOMNodelistLocal.item(i).text);
      EVALUATE(PrecioDEC,Precio);
      Producto."Unit Price":=ROUND(PrecioDEC,0.01);
    END;
    Producto."Price Includes VAT":=TRUE;
    DOMNodelistLocal:= XMLDomLocal.getElementsByTagName('ItemVendorNo');
    Producto."Vendor No.":=DOMNodelistLocal.item(i).text;
    DOMNodelistLocal:= XMLDomLocal.getElementsByTagName('ItemVendorItemNo');
    Producto."Vendor Item No.":=DOMNodelistLocal.item(i).text;
    DOMNodelistLocal:=
XMLDomLocal.getElementsByTagName('ItemGenProdPostingGroup');
    Producto."Gen. Prod. Posting Group":=DOMNodelistLocal.item(i).text;
    DOMNodelistLocal:=
XMLDomLocal.getElementsByTagName('ItemVATProdPostingGroup');
    Producto."VAT Prod. Posting Group":=DOMNodelistLocal.item(i).text;
XMLDomLocal.getElementsByTagName('ItemFechaUltimaModifTPV');
```

```
EVALUATE(FechaRegistro,COPYSTR(DOMNodelistLocal.item(i).text,9,2)+'/'+COPYSTR(D
OMNodelistLocal.item(i).text,6,2)+'/'+
            COPYSTR(DOMNodelistLocal.item(i).text,1,4));
      Producto."Last Date Modified":=FechaRegistro;
      Producto.MODIFY;
    END;

    i:=i+1;
    CodAnterior := mCod;
  UNTIL i= Cuantos;
  i:=0;
  CodAnterior := '';
  REPEAT
    DOMNodelistLocal:=
XMLDomLocal.getElementsByTagName('CodigosbarrasalternativosProducto');
    mCod:=DOMNodelistLocal.item(i).text;

    IF CodAnterior <> mCod THEN BEGIN
      DOMNodelistLocal:=
XMLDomLocal.getElementsByTagName('CodigosbarrasalternativosCodbarrasasociado');
      IF DOMNodelistLocal.length > 0 THEN BEGIN
        CodBarra := DOMNodelistLocal.item(i).text;
        CodBarraAlternativo.RESET;
        CodBarraAlternativo.SETRANGE(Producto, mCod);
        CodBarraAlternativo.DELETEALL;

        CodBarraAlternativo.RESET;
        CodBarraAlternativo.INIT;
        CodBarraAlternativo.Producto := mCod;
        CodBarraAlternativo."Cod. barras asociado" := CodBarra;
        CodBarraAlternativo.INSERT;
      END;
    END ELSE BEGIN
      DOMNodelistLocal:=
XMLDomLocal.getElementsByTagName('CodigosbarrasalternativosCodbarrasasociado');
      IF DOMNodelistLocal.length > 0 THEN BEGIN
        CodBarraAlternativo.RESET;
        CodBarraAlternativo.INIT;
        CodBarraAlternativo.Producto := mCod;
        CodBarra := DOMNodelistLocal.item(i).text;
        CodBarraAlternativo."Cod. barras asociado" := CodBarra;
        CodBarraAlternativo.INSERT;
      END;
    END;
    i:=i+1;
    CodAnterior := mCod;
  UNTIL i= CuantosBar;
  EXIT(Cuantos);
```

TPV_Sincronizar_Tarifa(ParProducto)

Esta función se invoca después de haber actualizado los datos del producto. En esta funcionalidad, lo que se actualiza son los precios de oferta en caso de tener o el precio de venta. Además, esta función se invoca para hacer la actualización diaria, ya que se ejecuta para cada producto modificado los días anteriores a la apertura.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si EncabezadoWeb.Vacio entonces
    Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_Sincronizar_Producto);
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXML('<TPV_Sincronizar_TarifaUnProducto xmlns=
                         http://tempuri.org/>'+'<Usuario>'+ConfigTpv.CodTienda
                         +'</Usuario>'+'<password>'+ConfigTpv.password+
                         '</password>'+'<Producto>'+ParProducto+'</Producto>'
                         +'<Tarifa>'+ConfigTpv.CodTarifaTpv+'</Tarifa>'+
                         '<TPV_Sincronizar_TarifaUnProducto>');
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>'' entonces
    Limpiar(EncabezadoWeb)
    Error('ERROR de conexión');
Si EstaVacío(NodoXml) entonces
    Crear(NodoXml)
NodoXml.Cargar(EncabezadoWeb)
NodoXml.Guardar('Tarifa.xml')
Tarifa se inicializa
Tarifa se filtra NoProd sea ParProducto
Si encuentra registro Tarifa entonces
   Borrar todos los registros
NodoLista=NodoXml.CargarElementos('PrecioVenta')
nvnumRegs=NodoLista.longitud
Si nvnumRegs=0 entonces
   Salir(0)
i=0
```

```
Repetir
   NodoLista=NodoXml.CogerElementos('PrecioVentaProducto')
   mCod=NodoLista.elemento(i).texto
  NodoLista=NodoXml.CogerElementos('PrecioVentaCodigo')
   mCod2=NodoLista.elemento(i).texto
   dvFechaInicio=0D
   NodoLista=NodoXml.CogerElementos('PrecioVentaFechaInicio')
   Evaluar(dvFechaInicio, CopiaCadena(NodoLista.elemento(i).texto,9,2)+'/'+
   CopiaCadena(NodoLista.elemento(i).texto,6,2)+'/'+
   CopiaCadena(NodoLista.elemento(i).texto,1,4))
   Si dvFechaInicio<>01010001D entonces
      Tarifa se inicializa
      Tarifa.NoProducto=ParProducto
      Tarifa.Tipo=GrupoPrecioCliente
      Tarifa.Codigo=ConfigTpv.CodTarifaTpv
      NodoLista=NodoXml.CogerElementos('PrecioVentaCodigoConcurrencia')
      Tarifa.CodigoConcurrencia=NodoLista.elemento(i).texto
     NodoLista=NodoXml.CogerElementos('PrecioVentaUnidadporCodigoMedida')
     Tarifa.UnidadPorCodigoMedida=NodoLista.elemento(i).texto
     NodoLista=NodoXml.CogerElementos('PrecioVentaTarifa')
     Si NodoLista.elemento(i).texto<>'' entonces
        Precio=FormatearImporteDec(NodoLista.elemento(i).texto)
        Evaluar(PrecioDEC,Precio)
        Tarifa.Tarifa=Redóndear(PrecioDEC,0.01)
     NodoLista=NodoXml.CogerElementos(PrecioVentaIncluyeIVA)
     Si NodoLista.elemento(i).texto=true entonces
        Tarifa.IncluyeIVA=Verdadero
     Sino
        Tarifa.IncluyeIVA=Falso
     NodoLista=NodoXml.CogerElementos(PrecioVentaDescuentoFactura)
     Si NodoLista.elemento(i).texto=true entonces
        Tarifa.DescuentoFactura=Verdadero
     Sino
        Tarifa.DescuentoFactura=Falso
     NodoLista=NodoXml.CogerElementos(PrecioVentaDescuentoLinea)
     Si NodoLista.elemento(i).texto=true entonces
        Tarifa.DescuentoLinea=Verdadero
     Sino
        Tarifa.DescuentoLinea=Falso
  NodoLista=NodoXml.CogerElementos('PrecioVentaGrupoNegocioIVA)
  Si NodoLista.elemento(i).texto<>'' entonces
      Tarifa.GrupoNegocioIVA=NodoLista.elemento(i).texto
  NodoLista=NodoXml.CogerElementos('PrecioVentaFechaUltMod)
  Evaluar(FechaRegistro, CopiaCadena(NodoLista.elemento(i).texto,9,2)+'/'+
   CopiaCadena(NodoLista.elemento(i).texto,6,2)+'/'+
   CopiaCadena(NodoLista.elemento(i).texto,1,4))
   Tarifa.FechaUltMod=FechaRegistro
   Tarifa.FechaInicio=dvFechaInicio
   dvFechaFin=0D
   NodoLista=NodoXml.CogerElementos('PrecioVentaFechaFin')
   Evaluar(dvFechaInicio, CopiaCadena(NodoLista.elemento(i).texto,9,2)+'/'+
   CopiaCadena(NodoLista.elemento(i).texto,6,2)+'/'+
   CopiaCadena(NodoLista.elemento(i).texto,1,4))
   Si dvFechaInicio<>01010001D entonces
```

```
      Tarifa.FechaFin=dvFechaFin
   Insertar registro Tarifa
 i=i+1
Hasta i=nvnumRegs
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());

IF Conf.FINDFIRST() THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
   END
ELSE
  ERROR('No Existe Configuración TPV');
BEGIN
 IF ISCLEAR(SoapHttpConnLocal) THEN
    CREATE(SoapHttpConnLocal);

 SoapHttpConnLocal.Property('EndPointURL', URLWS);
 SoapHttpConnLocal.Connect;
SoapHttpConnLocal.Property('SoapAction','http://tempuri.org/TPV_Sincronizar_TarifaUnProd
ucto');
 SoapHttpConnLocal.BeginMessage;
 IF ISCLEAR(SoapSerializeLocal) THEN
   CREATE(SoapSerializeLocal);
 SoapSerializeLocal.Init(SoapHttpConnLocal.InputStream);
 SoapSerializeLocal.StartEnvelope('','STANDARD','');
 SoapSerializeLocal.StartBody('');
 SoapSerializeLocal.WriteXML('<TPV_Sincronizar_TarifaUnProducto
xmlns="http://tempuri.org/">'+
                    '<Usuario>'+Conf."Cód. tienda"+'</Usuario>'+
                    '<password>'+Conf.Password+'</password>'+
                    '<Producto>'+PARProducto +'</Producto>'+
                    '<Tarifa>'+Conf."Cód. tarifa TPV"+'</Tarifa>'+
                     '</TPV_Sincronizar_TarifaUnProducto>');

 SoapSerializeLocal.EndBody;
 SoapSerializeLocal.EndEnvelope;
 SoapHttpConnLocal.EndMessage;
 IF  SoapHttpConnLocal.Error <> '' THEN BEGIN
    CLEAR(SoapHttpConnLocal);
    ERROR(Text0005);
 END;

 IF ISCLEAR(XMLDomLocal) THEN
   CREATE(XMLDomLocal);
 XMLDomLocal.async := FALSE;
 XMLDomLocal.load(SoapHttpConnLocal.OutputStream);
 XMLDomLocal.save(cduILR.DameDirXML() + 'tarifa1.xml'); // Esto es para que veas el
XML de Respuesta
```

```
  Tarifa.RESET;
  Tarifa.SETRANGE(Tarifa."Item No.",PARProducto);
  IF Tarifa.FINDFIRST() THEN
    Tarifa.DELETEALL();

  DOMNodelistLocal:= XMLDomLocal.getElementsByTagName('SalesPriceItemNo');
  nvNumRegs:= DOMNodelistLocal.length;
  IF nvNumRegs = 0 THEN
    EXIT(0);

  i:=0;
  REPEAT
    DOMNodelistLocal:= XMLDomLocal.getElementsByTagName('SalesPriceItemNo');
    mCod:=DOMNodelistLocal.item(i).text;

    DOMNodelistLocal:= XMLDomLocal.getElementsByTagName('SalesPriceSalesCode');
    mCod2:=DOMNodelistLocal.item(i).text;

    dvFechaInicio:=0D;
    DOMNodelistLocal:=     XMLDomLocal.getElementsByTagName('SalesPriceStartingDate');
EVALUATE(dvFechaInicio,COPYSTR(DOMNodelistLocal.item(i).text,9,2)+'/'+COPYSTR(D
OMNodelistLocal.item(i).text,6,2)+'/'+
          COPYSTR(DOMNodelistLocal.item(i).text,1,4));


IF dvFechaInicio <> 01010001D THEN
    BEGIN
      Tarifa.INIT;
      Tarifa."Item No.":=PARProducto;
      Tarifa."Sales Type":=Tarifa."Sales Type"::"Customer Price Group";
      Tarifa."Sales Code":=Conf."Cód. tarifa TPV";
      DOMNodelistLocal:=
XMLDomLocal.getElementsByTagName('SalesPriceCurrencyCode');
      Tarifa."Currency Code":=DOMNodelistLocal.item(i).text;
      DOMNodelistLocal:=
XMLDomLocal.getElementsByTagName('SalesPriceUnitofMeasureCode');
      Tarifa."Unit of Measure Code":=DOMNodelistLocal.item(i).text;
      DOMNodelistLocal:= XMLDomLocal.getElementsByTagName('SalesPriceUnitPrice');
      IF DOMNodelistLocal.item(i).text <> '' THEN
      BEGIN
        Precio:=FormatearImporteDEC(DOMNodelistLocal.item(i).text);
        EVALUATE(PrecioDEC,Precio);
        Tarifa."Unit Price":=ROUND(PrecioDEC,0.01);
      END;
      DOMNodelistLocal:=
XMLDomLocal.getElementsByTagName('SalesPricePriceIncludesVAT');
      IF DOMNodelistLocal.item(i).text='true' THEN
        Tarifa."Price Includes VAT":=TRUE
      ELSE
        Tarifa."Price Includes VAT":=FALSE;
      DOMNodelistLocal:=
XMLDomLocal.getElementsByTagName('SalesPriceAllowInvoiceDisc');
      IF DOMNodelistLocal.item(i).text='true' THEN
```

```
        Tarifa."Allow Invoice Disc.":=TRUE
      ELSE
        Tarifa."Allow Invoice Disc.":=FALSE;


      DOMNodelistLocal:=
XMLDomLocal.getElementsByTagName('SalesPriceAllowLineDisc');
      IF DOMNodelistLocal.item(i).text='true' THEN
        Tarifa."Allow Line Disc.":=TRUE
      ELSE
        Tarifa."Allow Line Disc.":=FALSE;
      DOMNodelistLocal:=
XMLDomLocal.getElementsByTagName('SalesPriceVATBusPostingGrPrice');
      IF DOMNodelistLocal.item(i).text <> '' THEN
      BEGIN
        Tarifa."VAT Bus. Posting Gr. (Price)":=DOMNodelistLocal.item(i).text;
      END;
      DOMNodelistLocal:=
XMLDomLocal.getElementsByTagName('SalesPriceUltimaFechaModificacion');
EVALUATE(FechaRegistro,COPYSTR(DOMNodelistLocal.item(i).text,9,2)+'/'+COPYSTR(D
OMNodelistLocal.item(i).text,6,2)+'/'+
            COPYSTR(DOMNodelistLocal.item(i).text,1,4));
      Tarifa."Últ. Fecha de modificación":=FechaRegistro;
      Tarifa."Starting Date":=dvFechaInicio;
      dvFechaFin:=0D;
      DOMNodelistLocal:= XMLDomLocal.getElementsByTagName('SalesPriceEndingDate');
EVALUATE(dvFechaFin,COPYSTR(DOMNodelistLocal.item(i).text,9,2)+'/'+COPYSTR(DO
MNodelistLocal.item(i).text,6,2)+'/'+
            COPYSTR(DOMNodelistLocal.item(i).text,1,4));
      IF dvFechaFin <> 01010001D THEN
        Tarifa."Ending Date":=dvFechaFin;
      Tarifa.INSERT;
    END;
    i:=i+1;
  UNTIL i= nvNumRegs;
  EXIT(DOMNodelistLocal.length);
```

FormatearImporteDec(ImporteNum)

Esta función sirve para convertir una variable de tipo texto en su correspondiente tipo numérico, para poder tratarla de forma correcta y evitar posibles errores.

PseudoCódigo

```
Si Posición(ImporteNum,'.')=0 entonces
   Si ImporteNum='' entonces
      Importe=0
   Sino
     Importe=ImporteNum
  Salir(Importe)
Sino
  Parte1=CopiaCadena(ImporteNum,1,Posicion(ImporteNum,'.')-1)
  Parte2=CopiaCadena(Posicion(ImporteNum,'.')+1,longitud(ImporteNum)
  Importe=Parte1+Parte2
  Salir(Importe)
```

Código Original

```
IF STRPOS(ImporteNum,'.')=0 THEN
  BEGIN
    IF ImporteNum='' THEN
      Importe:='0'
    ELSE
      Importe:=ImporteNum;
    EXIT(Importe);
  END
ELSE
  BEGIN
    request  := COPYSTR(ImporteNum, 1, STRPOS (ImporteNum, '.') - 1);
    request1 := COPYSTR(ImporteNum,STRPOS(ImporteNum,'.')+1,STRLEN(ImporteNum));
    Importe:=request +','+ request1;

    EXIT(Importe);
  END;
```

ComprobarERROR(NodoXml,TextoError)

Esta función se invoca para saber si la respuesta del servicio Web contiene error o muestra un mensaje indicando que se ha realizado correctamente.

PseudoCódigo

```
NodoLista=NodoXml.CogerElementos('Error')
Si NodoLista.Longitud>0 entonces
    TextoError=NodoLista.elemento(0).texto
     Salir(Verdadero)
Sino
     Salir(Falso)
```

Código Original

```
DOMNodelist:= XMLDom.getElementsByTagName('Error');

IF DOMNodelist.length > 0 THEN
BEGIN
  TxtError:=DOMNodelist.item(0).text;
  EXIT(TRUE);
END ELSE
  EXIT(FALSE);
```

FormatearImporte(ImporteNum)

Esta función sirve para dejar una variable con decimales en formato texto. La llamada al servicio Web se debe hacer con variables de formato texto

PseudoCódigo

```
Importe=Formatear(ImporteNum)
Si PosicionCadena(Importe,',')=0 entonces
   Si Longitud(Importe)>3 entonces
      Parte1=CopiaCadena(Importe,1,PosicionCadena(Importe,'.')-1)
```

```
      Parte2=CopiaCadena(Importe,Posicion(Importe,’.’)+1,Longitud(Importe))
      Importe=Parte1+Parte2
   Importe=Importe+’.00’;
Sino
   Parte1=CopiaCadena(Importe,1,PosicionCadena(Importe,’,’)-1)
   Si Longitud(Parte1)>3 entonces
      Parte2=CopiaCadena(Parte1,1,PosicionCadena(Parte1,’.’)-1)
      Parte3=CopiaCadena(Parte1,PosicionCadena(Parte1,’.’)+1,Longitud(Parte1)
      Parte1=Parte2+Parte3
  Auxiliar=CopiaCadena(Importe,Posicion(Importe,’,’)+1,Longitud(Importe))
  Si Longitud(Auxiliar)=1 entonces
      Auxiliar=Auxiliar+’0’
  Importe=Parte1+’.’+Auxiliar
Salir(Importe)
```

Código Original

```
Importe:=FORMAT(ImporteNum);
IF STRPOS(Importe,',')=0 THEN
  BEGIN
    IF STRLEN(Importe) > 3 THEN
      BEGIN
        request1 := COPYSTR (Importe, 1, STRPOS (Importe, '.') - 1);
        request2 := COPYSTR (Importe, STRPOS (Importe, '.') + 1, STRLEN (Importe));
        Importe:=request1+request2;
      END;
    Importe:=Importe+'.00';
  END
ELSE
  BEGIN
    request := COPYSTR (Importe, 1, STRPOS (Importe, ',') - 1);
    IF STRLEN(request) > 3 THEN
      BEGIN
        request1 := COPYSTR (request, 1, STRPOS (request, '.') - 1);
        request2 := COPYSTR (request, STRPOS (request, '.') + 1, STRLEN (request));
        request:=request1+request2;
      END;
    auxstring := COPYSTR (Importe, STRPOS (Importe, ',') + 1, STRLEN (Importe));
    IF STRLEN(auxstring)=1 THEN
      auxstring:=auxstring+'0';
    Importe:=request+'.'+auxstring;
  END;
EXIT(Importe);
```

CompruebaCliente(PARCliente)

Esta función se invoca cada vez que se registra una factura, para comprobar si el cliente está centralizado. En caso de no ser así, se debe dar de alta en CENTRAL.

PseudoCódigo

```
Si No TablaCliente.Coge(ParCliente) entonces
   Salir(Falso)
Si TablaCliente.EnviadoCentral entonces
   Salir(True)
numTpv=DameTPV()
```

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea numTpv
Si no encuentra registro ConfigTpv entonces
   Salir(Falso)
Si ConfigTpv.ModoTrabajo=On-Line entonces
   Si TPV_Alta_Clientes(ConfigTpv.CodTienda,ConfigTpv.Password, TablaCliente.No
      TablaCliente.Nombre,TablaCliente.Direccion,TablaCliente.CodigoPostal,
      TablaCliente.Ciudad,TablaCliente.Provincia,TablaCliente.telefono,
      TablaCliente.Movil,TablaCliente.Email,TablaCliente.NoRegIVA,
      TablaCliente.CodigoFormaPago,TablaCliente.Vendedor,
      TablaCliente.TerminosPago,TablaCliente.GrupoCliente,
      TablaCliente.GrupoRegistro,TablaCliente.GrupoRegistroIVA)) entonces
       Salir(Verdadero)
   Sino
       Salir(Falso)
```

Código Original

```
IF NOT rcdCliente.GET(PARCliente) THEN
  EXIT(FALSE);
IF rcdCliente."Enviado a CENTRAL" THEN
  EXIT(TRUE);

numTpv := cduILR.DameTPV();
Conf.RESET;
Conf.SETFILTER(Conf."Cód. TPV",numTpv);
IF NOT Conf.FINDFIRST() THEN
  EXIT(FALSE);
IF Conf."Modo de Trabajo"= Conf."Modo de Trabajo"::"On-Line" THEN
BEGIN
 IF (TPV_Alta_Clientes(Conf."Cód. tienda",Conf.Password,
                  rcdCliente."No.",rcdCliente.Name,rcdCliente.Address,
                  rcdCliente."Post Code",rcdCliente.City,
                  rcdCliente.County,rcdCliente."Phone No.",rcdCliente."Telex No.",
                  rcdCliente."E-Mail",
                  rcdCliente."VAT Registration No.",rcdCliente."Payment Method Code",
                  rcdCliente."Salesperson Code",rcdCliente."Payment Terms Code",
                  rcdCliente."Customer Posting Group",rcdCliente."Gen. Bus. Posting Group",
                  rcdCliente."VAT Bus. Posting Group")) THEN
   EXIT(TRUE)
 ELSE
   EXIT(FALSE);
END;
```

TPV_Transfer_Central(Usuario,Password,Ticket)

Esta función se invoca cuando se registra la transferencia en la tienda. Entonces, la función genera un fichero XML con todos los datos: cabecera y líneas, para que el servicio WEB pueda procesarlo y registrar la transferencia en CENTRAL para su posterior manipulación.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si ConfigTpv.ModoTrabajo=OffLine entonces
    Salir(Falso)
Si No ST_Abrir_Login(ConfigTpv.CodTienda,ConfigTpv.Password,Verdadero)
   Salir(Falso)
CabReposicion se inicializa
CabReposicion se filtra No sea Ticket
Si no encuentra registros CabReposicion entonces
   ERROR('Error, no encuentro la cabecera de reposicion')
FechaEnvio=Formateo(CabReposicion.FechaEnvio)
FechaRegistro=Formateo(CabReposicion.FechaRegistro)
FechaDoc=Formateo(CabReposicion.FechaCreacion)
Si EstaVacío(NodoXml) entonces
    Crear(NodoXml)
CadenaInicial='<NewDataSet>'
CadenaFinal='</NewDataSet>'
LinReposicion se inicializa
LinReposicion se filtra NoDocumento sea Ticket
Si encuentra registros LinReposicion entonces
   i=1
  Repetir
      Cadena(i)=Cadena(i)+'<Lineas>'
      Cadena(i)=Cadena(i)+'<NumLinea>'+Formateo(LinReposicion.NoLinea)+
      '</NumLinea>'
      Cadena(i)=Cadena(i)+'<No.>'+LinReposicion.No+'</No.>'
      Cadena(i)=Cadena(i)+'<Quantity>'+Formateo(LinReposicion.Cantidad)+
      '</Quantity>'
      Cadena(i)=Cadena(i)+'<AlmOrigen>'+LinReposicion.AlmacenOrigen+
      '</AlmOrigen>'
      Cadena(i)=Cadena(i)+'<AlmDestino>'+LinReposicion.AlmacenDestino+
      '</AlmDestino>'
      Cadena(i)=Cadena(i)+'<TipoDestino>'+Formateo(LinReposicion.TipoDestino)+
      '</TipoDestino>'
      Cadena(i)=Cadena(i)+'</Lineas>'
      i=i+1
  Hasta siguiente registro LinReposicion sea vacío
Si EstaVacío(EncabezadoWeb) entonces
    Crear(EncabezadoWeb)
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_Transfer_Central');
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
```

```
ConectorWeb.InicioEstructura(‘’,’STANDARD’,’’)
ConectorWeb.InicioCuerpo(‘’);
ConectorWeb.EscribirXML(‘<TPV_Transfer_Central xmlns=http://tempuri.org/>’+
                  ‘<Usuario>’+Usuario+’</Usuario>’+’<Password>’+Password+
                  ’</Password>’+’<NumeroTicket>’+Ticket+’</NumeroTicket>’+
                  ‘<AlmOrigen>’+CabReposicion.AlmacenOrigen+’</AlmOrigen>’+
                  ‘<AlmDestino>’+CabReposicion.Destino+’</AlmDestino>’+
                  ‘<FechaEnvio>’+Fomateo(CabReposicion.FechaEnvio)+
                  ’</FechaEnvio>’+’<FechaReg>’+
                  Fomateo(CabReposicion.FechaRegistro)+’</FechaReg>’+’<FechaDoc>’
                  +Formateo(CabReposicion.FechaCreacion)+’</FechaDoc>’+
                  ‘<Transportista>’+Formateo(CabReposicion.Transportista)+
               ‘</Transportista>’+’<DatosLineasStr>’);
ConectorWeb.EscribirCadena(CadenaInicial)
x=1
Repetir
   ConectorWeb.EscribirCadena(Cadena(x))
   x=x+1
Hasta x>i
ConectorWeb.EscribirCadena(CadenaFinal)
ConectorWeb.EscribirXML(‘</DatosLineasStr>’’</TPV_Transfer_Central>’)
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>’’ entonces
   Salir(Falso)
Si EstaVacío(NodoXml) entonces
   Crear(NodoXml)
NodoXml.Cargar(EncabezadoWeb)
NodoXml.Guardar(‘Transfer.Xml’)
Si ComprobarERROR(NodoXml,TxtError) entonces
   ConfigTpv.NoFallos=ConfigTpv.NoFallos+1
   Modificar registro
   Si ConfigTpv.NoFallos>3 entonces
      ConfigTpv.FechaOffLine=Hoy
      ConfigTpv.HoraOffLine=Hoy
      ConfigTpv.ModoTrabajo=OffLine
      Modificar registro
  Salir(Falso)
ConfigTpv.NoFallos=0
Salir(Verdadero)
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF Conf.FINDFIRST() THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
   END
ELSE
  ERROR('No Existe Configuración TPV');
```

```
IF Conf."Modo de Trabajo" = Conf."Modo de Trabajo"::"Off-Line" THEN
  EXIT(FALSE);
IF NOT ST_Abrir_Login(Conf."Cód. tienda",Conf.Password,TRUE) THEN
  EXIT(FALSE);
CabRep.RESET;
CabRep.SETRANGE("Nº", Ticket);
IF NOT CabRep.FINDFIRST() THEN
  ERROR('ERROR no encuentro la cabecera de reposición.',Ticket);

FechaEnvio        := FORMAT(CabRep."Fecha envío");
FechaRegistro     := FORMAT(CabRep."Fecha registro");
FechaDoc          := FORMAT(CabRep."Fecha creación");

 BEGIN
   IF ISCLEAR(XMLDom) THEN
     CREATE (XMLDom);

   CadenaInicial:='<NewDataSet>';
   CadenaFinal:='</NewDataSet>';
   LinRep.RESET;
   LinRep.SETRANGE(LinRep."Nº documento",Ticket);
   IF LinRep.FINDSET() THEN
   BEGIN
     i:=1;
    REPEAT
       Cadena[i] := Cadena[i] + '<Lineas>';
       Cadena[i] := Cadena[i] + '<NumLinea>'+FORMAT(LinRep."Nº línea")+ '</NumLinea>';
       Cadena[i] := Cadena[i] + '<No.>'+LinRep."Nº"+ '</No.>';
       Cadena[i] := Cadena[i] + '<Quantity>'+FORMAT(LinRep.Cantidad)+'</Quantity>';
       Cadena[i] := Cadena[i] + '<AlmOrigen>'+LinRep."Almacen Origen"+'</AlmOrigen>';
       Cadena[i] := Cadena[i] + '<AlmDestino>'+LinRep."Almacen Destino"+'</AlmDestino>';
       Cadena[i]        :=        Cadena[i]        +        '<TipoDestino>'+FORMAT(LinRep."Tipo
Destino")+'</TipoDestino>';
       Cadena[i] := Cadena[i] + '</Lineas>';
       i:=i+1;
    UNTIL LinRep.NEXT=0;
   END;
   IF ISCLEAR(SoapHttpConn) THEN
     CREATE(SoapHttpConn);

   SoapHttpConn.Property('EndPointURL', URLWS);
   SoapHttpConn.Connect;

   SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_Transfer_Central');
   SoapHttpConn.BeginMessage;
   IF ISCLEAR(SoapSerialize) THEN
     CREATE(SoapSerialize);
   SoapSerialize.Init(SoapHttpConn.InputStream);
   SoapSerialize.StartEnvelope('','STANDARD','');
   SoapSerialize.StartBody('');
   SoapSerialize.WriteXML('<TPV_Transfer_Central xmlns="http://tempuri.org/">'+
                    '<Usuario>'+Usuario+'</Usuario>'+
                    '<password>'+Password+'</password>'+
```

```
                '<NumeroTicket>'+Ticket+'</NumeroTicket>'+
                '<AlmOrigen>' + CabRep."Almacen Origen" + '</AlmOrigen>' +
                '<AlmDestino>' + CabRep.Destino + '</AlmDestino>' +
                '<FechaEnvio>' + FORMAT(CabRep."Fecha envío") + '</FechaEnvio>' +
                '<FechaReg>' + FORMAT(CabRep."Fecha registro") + '</FechaReg>' +
                '<FechaDoc>' + FORMAT(CabRep."Fecha creación") + '</FechaDoc>' +
                '<Transportista>' + FORMAT(CabRep.Tansportista) + '</Transportista>' +
                '<DatosLineasStr>');

SoapSerialize.WriteString(CadenaInicial);
x:=1;
REPEAT
  SoapSerialize.WriteString(Cadena[x]);
  x:=x+1;
UNTIL x > i;
SoapSerialize.WriteString(CadenaFinal);
SoapSerialize.WriteXML('</DatosLineasStr></TPV_Transfer_Central>');
SoapSerialize.EndBody;
SoapSerialize.EndEnvelope;
SoapHttpConn.EndMessage;

IF SoapHttpConn.Error <> '' THEN
   EXIT(FALSE);

IF ISCLEAR(XMLDom) THEN
   CREATE(XMLDom);

XMLDom.async := FALSE;
XMLDom.load(SoapHttpConn.OutputStream);
XMLDom.save(cduILR.DameDirXML() + 'transfer.xml');
IF ComprobarERROR(XMLDom,TxtError) THEN
BEGIN
  Conf."Nº de fallos" +=1;
  Conf.MODIFY;
  IF Conf."Nº de fallos" > 3 THEN BEGIN
     Conf."Fecha entrada Off-Line":=WORKDATE;
     Conf."Hora Entrada Off-Line":=TIME;
     Conf."Modo de Trabajo":=Conf."Modo de Trabajo"::"Off-Line";
     Conf.MODIFY;
  END;
  EXIT(FALSE);
END;
Conf."Nº de fallos" :=0;
Conf.MODIFY;
EXIT(TRUE);
```

TPV_Login_Coordinador(Coord,CoordPwd,OcultarError,NombreCoord)

Esta función se invoca cuando se quiere registrar una transferencia de coordinador, que el sistema obliga a introducir usuario y password, para certificar que la persona que se lleva los productos es la responsable de coordinar la tienda y evitar posibles

manipulaciones de producto indebidas. Para ello, se conecta a la CENTRAL a buscar el usuario introducido y comprobar que los datos son correctos.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si EncabezadoWeb.Vacio entonces
    Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_Login_Coordinador');
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXml('<TPV_Login_Coordinador xmlns=http://tempuri.org/>
                 <UserID>'+ConfigTpv.CodTienda+'</UserID>'+'<password>'+
                ConfigTpv.Password+'</Password>'+'<CoordId>'+Coord+'</CoordId>'
                +'<pwdCoord>'+CoordPwd+'</pwdCorrd>'+
                '<TPV_Login_Coordinador>')
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>'' entonces
   Salir(Falso)
NodoXml.Cargar(EncabezadoWeb)
NodoXml.Guardar('Login_Coordinador.xml')
Si ComprobarERROR(NodoXml,TxtError) entonces
   Salir(Falso)
NodoLista=NodoXml.CogerElementos(MobileUser)
Si NodoLista.Longitud=0 entonces
   Salir(Falso)
Si NodoLista.elemento(0).texto='' entonces
   Salir(Falso)
NombreCoord=NodoLista.elemento(0).texto
Salir(Verdadero)
```

Código Original

```
NombreCoord:='';
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF Conf.FINDFIRST THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
```

```
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
   END
ELSE
  ERROR('No Existe Configuración TPV');

CLEAR(SoapSerializeLocal);
CLEAR(SoapHttpConnLocal);
CLEAR(DOMNodelistLocal);
CLEAR(XMLDomLocal);
CREATE(SoapHttpConnLocal);
CREATE(SoapSerializeLocal);
CREATE(XMLDomLocal);

SoapHttpConnLocal.Property('EndPointURL', URLWS);
SoapHttpConnLocal.Connect;
SoapHttpConnLocal.Property('SoapAction','http://tempuri.org/TPV_Login_Coordinador');
SoapHttpConnLocal.BeginMessage;
SoapSerializeLocal.Init(SoapHttpConnLocal.InputStream);
SoapSerializeLocal.StartEnvelope('','STANDARD','');
SoapSerializeLocal.StartBody('');
SoapSerializeLocal.WriteXML('<TPV_Login_Coordinador
xmlns="http://tempuri.org/"><UserId>'+Conf."Cód. tienda" +'</UserId>'+
                                    '<Password>'+ Conf.Password+'</Password>' +
                                    '<CoordId>' + xCoordID + '</CoordId>'+
                                    '<pwdCoord>' +xCoordPwd + '</pwdCoord>' +
                                    '</TPV_Login_Coordinador>');


SoapSerializeLocal.EndBody;
SoapSerializeLocal.EndEnvelope;
SoapHttpConnLocal.EndMessage;

IF SoapHttpConnLocal.Error <> '' THEN
  EXIT(FALSE);

XMLDomLocal.async := FALSE;
XMLDomLocal.load(SoapHttpConnLocal.OutputStream);
XMLDomLocal.save(cduILR.DameDirXML() + 'Login_Coordinador.xml');
IF ComprobarERROR(XMLDomLocal,TxtError) THEN
  EXIT(FALSE);

DOMNodelistLocal:= XMLDomLocal.getElementsByTagName('MobileUsersUserName');

IF DOMNodelistLocal.length = 0 THEN
  EXIT(FALSE);

IF DOMNodelistLocal.item(0).text = '' THEN
  EXIT(FALSE);
NombreCoord:=DOMNodelistLocal.item(0).text;

EXIT(TRUE);
```

TPV_Transfer_Coordinador(Usuario,Password,Ticket)

Esta función se invoca cuando se registra la transferencia a coordinador, para que esos movimientos de productos provocados por la transferencia queden reflejados en CENTRAL y se tenga el stock actualizado al instante.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si ConfigTpv.ModoTrabajo=OffLine entonces
    Salir(Falso)
Si No ST_Abrir_Login(ConfigTpv.CodTienda,ConfigTpv.Password,Verdadero)
   Salir(Falso)
CabRep se inicializa
CabRep se filtra No sea Ticket
Si no encuentra registros CabRep entonces
    ERROR('Error no encuentro la cabecera de reposicion')
FechaEnvio=Formateo(CabRep.FechaEnvio);
FechaRegistro=Formateo(CabRep.FechaRegistro)
FechaDoc=Formateo(CabRep.FechCreacion)
Si EstaVacío(NodoXml) entonces
   Crear(NodoXml)
CadenaInicial='<NewDataSet>'
CadenaFinal='</NewDataSet>'
LineaRep se inicializa
LineaRep se filtra NoDocumento sea Ticket
Si encuentra registros LineaRep entonces
   i=1
   Repetir
      Cadena(i)=Cadena(i)+'<Lineas>'
      Cadena(i)=Cadena(i)+'<NumLinea>'+Formateo(LineaRep.NoLinea)+
      '</NumLinea>'
      Cadena(i)=Cadena(i)+'<No.>'+LineaRep.No+'</No.>'
      Cadena(i)=Cadena(i)+'<Quantity>'+Formateo(LineaRep.Cantidad)+
      '</Quantity>'
      Cadena(i)=Cadena(i)+'<AlmOrigen>'+LineaRep.AlmacenOrigen+</AlmOrigen>
      Cadena(i)=Cadena(i)+'<AlmDestino>'+LineaRep.AlmacenOrigen+
      '</AlmDestino>'
     Cadena(i)=Cadena(i)+'</Lineas>'
       i=i+1
   Hasta siguiente registro LineaRep sea vacío
Si EstaVacío(EncabezadoWeb) entonces
   Crear(EncabezadoWeb)
```

```
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_Transfer_Coordinador);
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXml('<TPV_Transfer_Coordinador xmlns=http://tempuri.org/>'+
                '<Usuario>'+Usuario+'</Usuario>'+'<password>'+Password+
                '</password>'+'<NumeroTicket>'+Ticket+'</NumeroTicket>'+
                '<AlmOrigen>'+CabRep.AlmacenOrigen+'</AlmOrigen>'+
               '<AlmDestino>'+CabRep.Destino+'</AlmDestino>'+'<FechaEnvio>'+
               Formateo(CabRep.FechaEnvio)+'</FechaEnvio>'+'<FechaReg>'+
               Formateo(CabRep.FechaRegistro)+'</FechaReg>'+'<FechaDoc>'+
               Formateo(CabRep.FechaCreacion)+'</FechaDoc>'+'<DatosLineasStr>')
ConectorWeb.EscribirCadena(CadenaInicial)
x=1
Repetir
    ConectorWeb.EscribirCadena(Cadena(i))
    x=x+1
Hasta x>i
ConectorWeb.EscribirCadena(CadenaFinal)
ConectorWeb.EscribirXml('</DatosLineasStr></TPV_Transfer_Coordinador>');
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>'' entonces
       Salir(Falso)
Si EstaVacío(NodoXml)
    Crear(NodoXml)
NodoXml.Cargar(EncabezadoWeb)
NodoXml.Guardar('transfer.xml')
Si ComprobarERROR(NodoXml,TxtError) entonces
   ConfigTpv.NoFallos=ConfigTpv.NoFallos+1
   Modificar registro
   Si ConfigTpv.NoFallos>3 entonces
      ConfigTpv.FechaOffLine=Hoy
      ConfigTpv.HoraOffLine=Hoy
      ConfigTpv.ModoTrabajo=OffLine
      Modificar registro
 Salir(Falso)
ConfigTpv.NoFallos=0
Salir(Verdadero)
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF Conf.FINDFIRST THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
```

```
      ERROR('No existe Configuración del Servicio WEB');
    END
ELSE
  ERROR('No Existe Configuración TPV');
IF Conf."Modo de Trabajo" = Conf."Modo de Trabajo"::"Off-Line" THEN
  EXIT(FALSE);
IF NOT ST_Abrir_Login(Conf."Cód. tienda",Conf.Password,TRUE) THEN
  EXIT(FALSE);

CabRep.RESET;
CabRep.SETRANGE("Nº", Ticket);
IF NOT CabRep.FINDFIRST() THEN
  ERROR('ERROR no encuentro la cabecera de reposición.',Ticket);
FechaEnvio        := FORMAT(CabRep."Fecha envío");
FechaRegistro     := FORMAT(CabRep."Fecha registro");
FechaDoc          := FORMAT(CabRep."Fecha creación");

    IF ISCLEAR(XMLDom) THEN
      CREATE (XMLDom);

    CadenaInicial:='<NewDataSet>';
    CadenaFinal:='</NewDataSet>';
    LinRep.RESET;
    LinRep.SETRANGE(LinRep."Nº documento",Ticket);
    IF LinRep.FINDSET() THEN
    BEGIN
      i:=1;
     REPEAT
        Cadena[i] := Cadena[i] + '<Lineas>';
        Cadena[i] := Cadena[i] + '<NumLinea>'+FORMAT(LinRep."Nº línea")+ '</NumLinea>';
        Cadena[i] := Cadena[i] + '<No.>'+LinRep."Nº"+ '</No.>';
        Cadena[i] := Cadena[i] + '<Quantity>'+FORMAT(LinRep.Cantidad)+'</Quantity>';
        Cadena[i] := Cadena[i] + '<AlmOrigen>'+LinRep."Almacen Origen"+'</AlmOrigen>';
        Cadena[i] := Cadena[i] + '<AlmDestino>'+LinRep."Almacen Destino"+'</AlmDestino>';
        Cadena[i] := Cadena[i] + '</Lineas>';
        i:=i+1;
     UNTIL LinRep.NEXT=0;
    END;
    IF ISCLEAR(SoapHttpConn) THEN
      CREATE(SoapHttpConn);

    SoapHttpConn.Property('EndPointURL', URLWS);
    SoapHttpConn.Connect;
SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_Transfer_Coordinador');
    SoapHttpConn.BeginMessage;
    IF ISCLEAR(SoapSerialize) THEN
      CREATE(SoapSerialize);
    SoapSerialize.Init(SoapHttpConn.InputStream);
    SoapSerialize.StartEnvelope('','STANDARD','');
    SoapSerialize.StartBody('');
    SoapSerialize.WriteXML('<TPV_Transfer_Coordinador xmlns="http://tempuri.org/">'+
                    '<Usuario>'+Usuario+'</Usuario>'+
                    '<password>'+Password+'</password>'+
                    '<NumeroTicket>'+Ticket+'</NumeroTicket>'+
```

```
                '<AlmOrigen>' + CabRep."Almacen Origen" + '</AlmOrigen>' +
                '<AlmDestino>' + CabRep.Destino + '</AlmDestino>' +
                '<FechaEnvio>' + FORMAT(CabRep."Fecha envío") + '</FechaEnvio>' +
                '<FechaReg>' + FORMAT(CabRep."Fecha registro") + '</FechaReg>' +
                '<FechaDoc>' + FORMAT(CabRep."Fecha creación") + '</FechaDoc>' +
                '<DatosLineasStr>');
SoapSerialize.WriteString(CadenaInicial);
x:=1;


REPEAT
  SoapSerialize.WriteString(Cadena[x]);
  x:=x+1;
UNTIL x > i;
SoapSerialize.WriteString(CadenaFinal);
SoapSerialize.WriteXML('</DatosLineasStr></TPV_Transfer_Coordinador>');
SoapSerialize.EndBody;
SoapSerialize.EndEnvelope;
SoapHttpConn.EndMessage;

IF SoapHttpConn.Error <> '' THEN
   EXIT(FALSE);

IF ISCLEAR(XMLDom) THEN
   CREATE(XMLDom);

XMLDom.async := FALSE;
XMLDom.load(SoapHttpConn.OutputStream);
XMLDom.save(cduILR.DameDirXML() + 'transfer.xml');

IF ComprobarERROR(XMLDom,TxtError) THEN
BEGIN
  Conf."Nº de fallos" +=1;
  Conf.MODIFY;
  IF Conf."Nº de fallos" > 3 THEN BEGIN
     Conf."Fecha entrada Off-Line":=WORKDATE;
     Conf."Hora Entrada Off-Line":=TIME;
     Conf."Modo de Trabajo":=Conf."Modo de Trabajo"::"Off-Line";
     Conf.MODIFY;
  END;
  EXIT(FALSE);
END;

Conf."Nº de fallos" :=0;
Conf.MODIFY;
EXIT(TRUE);
```

TPV_Transfer_Robos(Usuario,Password,Ticket)

Esta función se invoca para cuando se registra una serie de productos que han sido robados de la tienda y así dar de baja del stock de la misma y los movimientos de producto queden reflejados en CENTRAL.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si ConfigTpv.ModoTrabajo=OffLine entonces
    Salir(Falso)
Si No ST_Abrir_Login(ConfigTpv.CodTienda,ConfigTpv.Password,Verdadero)
   Salir(Falso)
CabRep se inicializa
CabRep se filtra No sea Ticket
Si no encuentra registros CabRep entonces
    ERROR('Error no encuentro la cabecera de reposicion')
FechaRegistro=Formateo(CabRep.FechaRegistro)
Si EstaVacío(NodoXml) entonces
   Crear(NodoXml)
CadenaInicial='<NewDataSet>'
CadenaFinal='</NewDataSet>'
LineaRep se inicializa
LineaRep se filtra NoDocumento sea Ticket
Si encuentra registros LineaRep entonces
   i=1
   Repetir
      Cadena(i)=Cadena(i)+'<Lineas>'
      Cadena(i)=Cadena(i)+'<No.>'+LineaRep.No+'</No.>'
      Cadena(i)=Cadena(i)+'<Quantity>'+Formateo(LineaRep.Cantidad)+
      '</Quantity>'
      Cadena(i)=Cadena(i)+'<AlmOrigen>'+LineaRep.AlmacenOrigen+</AlmOrigen>
      Cadena(i)=Cadena(i)+'<AlmDestino>'+LineaRep.AlmacenOrigen+
      '</AlmDestino>'
     Cadena(i)=Cadena(i)+'</Lineas>'
       i=i+1
   Hasta siguiente registro LineaRep sea vacío
Si EstaVacío(EncabezadoWeb) entonces
   Crear(EncabezadoWeb)
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_Transfer_Robos);
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXml('<TPV_Transfer_Robos xmlns=http://tempuri.org/>'+
                  '<Usuario>'+Usuario+'</Usuario>'+'<password>'+Password+
                  '</password>'+'<NumeroTicket>'+Ticket+'</NumeroTicket>'+
                  '<AlmOrigen>'+CabRep.AlmacenOrigen+'</AlmOrigen>'+
                 '<AlmDestino>'+CabRep.Destino+'</AlmDestino>'+''<FechaReg>'+
```

```
                Formateo(CabRep.FechaRegistro)+'</FechaReg>'+<DatosLineasStr>')
ConectorWeb.EscribirCadena(CadenaInicial)
x=1
Repetir
    ConectorWeb.EscribirCadena(Cadena(i))
    x=x+1
Hasta x>i
ConectorWeb.EscribirCadena(CadenaFinal)
ConectorWeb.EscribirXml('</DatosLineasStr></TPV_Transfer_Robos>');
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>'' entonces
       Salir(Falso)
Si EstaVacío(NodoXml)
    Crear(NodoXml)
NodoXml.Cargar(EncabezadoWeb)
NodoXml.Guardar('transfer.xml')
Si ComprobarERROR(NodoXml,TxtError) entonces
   ConfigTpv.NoFallos=ConfigTpv.NoFallos+1
   Modificar registro
   Si ConfigTpv.NoFallos>3 entonces
      ConfigTpv.FechaOffLine=Hoy
      ConfigTpv.HoraOffLine=Hoy
      ConfigTpv.ModoTrabajo=OffLine
      Modificar registro
 Salir(Falso)
ConfigTpv.NoFallos=0
Salir(Verdadero)
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF Conf.FINDFIRST THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
   END
ELSE
  ERROR('No Existe Configuración TPV');



IF Conf."Modo de Trabajo" = Conf."Modo de Trabajo"::"Off-Line" THEN
  EXIT(FALSE);

IF NOT ST_Abrir_Login(Conf."Cód. tienda",Conf.Password,TRUE) THEN
  EXIT(FALSE);

CabRep.RESET;
CabRep.SETRANGE("Nº", Ticket);
IF NOT CabRep.FINDFIRST() THEN
```

```
  ERROR('ERROR no encuentro la cabecera de reposición.',Ticket);

FechaRegistro      := FORMAT(CabRep."Fecha registro");
    IF ISCLEAR(XMLDom) THEN
      CREATE (XMLDom);

    CadenaInicial:='<NewDataSet>';
    CadenaFinal:='</NewDataSet>';

    LinRep.RESET;
    LinRep.SETRANGE(LinRep."Nº documento",Ticket);
    IF LinRep.FINDSET() THEN
    BEGIN
      i:=1;
     REPEAT
        Cadena[i] := Cadena[i] + '<Lineas>';
        Cadena[i] := Cadena[i] + '<No.>'+LinRep."Nº"+ '</No.>';
        Cadena[i] := Cadena[i] + '<Quantity>'+FORMAT(LinRep.Cantidad)+'</Quantity>';
        Cadena[i] := Cadena[i] + '<AlmOrigen>'+LinRep."Almacen Origen"+'</AlmOrigen>';
        Cadena[i] := Cadena[i] + '<AlmDestino>'+LinRep."Almacen Destino"+'</AlmDestino>';
        Cadena[i] := Cadena[i] + '</Lineas>';
        i:=i+1;
     UNTIL LinRep.NEXT=0;
    END;
    IF ISCLEAR(SoapHttpConn) THEN
      CREATE(SoapHttpConn);

    SoapHttpConn.Property('EndPointURL', URLWS);
    SoapHttpConn.Connect;
    SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_Transfer_Robos');
    SoapHttpConn.BeginMessage;
    IF ISCLEAR(SoapSerialize) THEN
      CREATE(SoapSerialize);
    SoapSerialize.Init(SoapHttpConn.InputStream);
    SoapSerialize.StartEnvelope('','STANDARD','');
    SoapSerialize.StartBody('');
    SoapSerialize.WriteXML('<TPV_Transfer_Robos xmlns="http://tempuri.org/">'+
                           '<Usuario>'+Usuario+'</Usuario>'+
                           '<password>'+Password+'</password>'+
                           '<NumeroTicket>'+Ticket+'</NumeroTicket>'+
                           '<AlmOrigen>' + CabRep."Almacen Origen" + '</AlmOrigen>' +
                           '<AlmDestino>' + CabRep.Destino + '</AlmDestino>' +
                           '<FechaReg>' + FORMAT(CabRep."Fecha registro") + '</FechaReg>' +
                           '<DatosLineasStr>');

    SoapSerialize.WriteString(CadenaInicial);
    x:=1;
    REPEAT
     SoapSerialize.WriteString(Cadena[x]);
     x:=x+1;
    UNTIL x > i;
    SoapSerialize.WriteString(CadenaFinal);
    SoapSerialize.WriteXML('</DatosLineasStr></TPV_Transfer_Robos>');
    SoapSerialize.EndBody;
```

```
SoapSerialize.EndEnvelope;
SoapHttpConn.EndMessage;

IF SoapHttpConn.Error <> '' THEN
  EXIT(FALSE);

IF ISCLEAR(XMLDom) THEN
  CREATE(XMLDom);

XMLDom.async := FALSE;
XMLDom.load(SoapHttpConn.OutputStream);
XMLDom.save(cduILR.DameDirXML() + 'transfer.xml');

IF ComprobarERROR(XMLDom,TxtError) THEN
BEGIN
 Conf."Nº de fallos" +=1;
 Conf.MODIFY;
 IF Conf."Nº de fallos" > 3 THEN BEGIN
   Conf."Fecha entrada Off-Line":=WORKDATE;
   Conf."Hora Entrada Off-Line":=TIME;
   Conf."Modo de Trabajo":=Conf."Modo de Trabajo"::"Off-Line";
   Conf.MODIFY;
 END;
 EXIT(FALSE);
END;
Conf."Nº de fallos" :=0;
Conf.MODIFY;
EXIT(TRUE);
```

TPV_Coordinador_BuscaTransfer(NumeroTicket,Coordinador)

Esta función se invoca cuando el coordinador quiere dejar mercancía en otra tienda. El sistema busca en CENTRAL la transferencia que ha indicado el coordinador, para traerse las líneas y mostrar cada una de ellas para que el coordinador elija que productos y la cantidad de los mismos quiere dejar en la tienda.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si EncabezadoWeb.Vacio entonces
    Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/
                    TPV_Coordinador_BuscaTransfer');
```

```
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura(‘’,’STANDARD’,’’)
ConectorWeb.InicioCuerpo(‘’);
ConectorWeb.EscribirXml(‘<TPV_Coordinador_BuscaTransfer xmlns=
                http://tempuri.org/”>’+’<Usuario>’+ConfigTpv.CodTienda+</Usuario>
                +’<password>’+ConfigTpv.password+’</password>’+’<NumeroTicket>
                +Ticket+’</NumeroTicket>’+’<Coordinador>’+Coordinador+
                ‘</Coordinador>’+’</TPV_Coordinador_BuscaTransfer>’)
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>’’ entonces
   Vaciar(EncabezadoWeb)
   ERROR(‘Error al traer la transferencia’)
Si EstaVacío(NodoXml) entonces
   Crear(NodoXml)
NodoXml.Cargar(EncabezadoWeb)
NodoXml.Guardar(‘transfercoord.xml’)
NodoLista=NodoXml.CogerElementos(‘LineaTransferEnvio)
Si NodoLista.longitud=0 entonces
   Salir(0)
Sino
   Cuantos=NodoLista.longitud
txtAux=NodoLista.elemento(i).texto
CabTransfer se inicializa
CabTransfer se filtra No sea txtAux
Si no encuentra registro CabTransfer entonces
   CabTransfer se inicializa
   CabTransfer.Tipo=Transferencia Coordinador
   CabTransfer.No=txtAux
   CabTransfer.Almacen Origen=Coordinador
   CabTransfer.Destino=ConfigTpv.CodAlmacen
   Insertar registro CabTransfer
LinTransfer se inicializa
LinTransfer se filtra NoDoc sea txtAux
Si encuentra registros LinTransfer entonces
   Borrar todos los registros LinTransfer
i=0
Repetir
   LinTransfer se inicializa
   LinTransfer.NoDoc=CabTransfer.No
   NodoLista=NodoXml.CogerElementos(‘TransferEnvioNoLinea)
       Si NodoLista.Longitud>0 entonces
          txtAux=NodoLista.elemento(i).texto
       Evaluar(LinTransfer.NoLinea,txtAux)
 NodoLista=NodoXml.CogerElementos(‘TransferEnvioNoProducto)
       Si NodoLista.Longitud>0 entonces
          txtAux=NodoLista.elemento(i).texto
       LinTransfer.No=txtAux
 NodoLista=NodoXml.CogerElementos(‘TransferEnvioCantidad)
       Si NodoLista.Longitud>0 entonces
```

```
          txtAux=NodoLista.elemento(i).texto
        Evaluar(LinTransfer.Cantidad,txtAux)
  NodoLista=NodoXml.CogerElementos('TransferEnvioDescripcion)
        Si NodoLista.Longitud>0 entonces
          txtAux=NodoLista.elemento(i).texto
       LinTransfer.Descripcion=txtAux
  NodoLista=NodoXml.CogerElementos('TransferEnvioCantidad)
        Si NodoLista.Longitud>0 entonces
          txtAux=NodoLista.elemento(i).texto
        Evaluar(LinTransfer.Cantidad,txtAux)
  NodoLista=NodoXml.CogerElementos('TransferEnvioCantidadConsumida)
        Si NodoLista.Longitud>0 entonces
          txtAux=NodoLista.elemento(i).texto
        Evaluar(LinTransfer.CantidadEnviada,txtAux)
LinTransfer.AlmacenOrigen=CabTransfer.AlmacenOrigen
LinTransfer.AlmacenDestino=CabTransfer.Destino
Insertar registro LinTransfer
Hasta i=Cuantos
Salir(Cuantos)
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF NOT Conf.FINDFIRST() THEN
  ERROR('No Existe Configuración TPV');

IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
  URLWS:=Conf."URL Servicio WEB CENTRAL"
ELSE
  ERROR('No existe Configuración del Servicio WEB');

IF ISCLEAR(SoapHttpConn) THEN
  CREATE(SoapHttpConn);

SoapHttpConn.Property('EndPointURL', URLWS);
SoapHttpConn.Connect;
SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_Coordinador_BuscaTransfer');
SoapHttpConn.BeginMessage;
IF ISCLEAR(SoapSerialize) THEN
  CREATE(SoapSerialize);
SoapSerialize.Init(SoapHttpConn.InputStream);
SoapSerialize.StartEnvelope('','STANDARD','');
SoapSerialize.StartBody('');

SoapSerialize.WriteXML('<TPV_Coordinador_BuscaTransfer xmlns="http://tempuri.org/">'+
                       '<Usuario>'+Conf."Cód. tienda"+'</Usuario>'+
                       '<password>'+Conf.Password+'</password>'+
                       '<NumeroTicket>'+NumeroTicket +'</NumeroTicket>'+
                       '<Coordinador>'+Coordinador+'</Coordinador>'+
                       '</TPV_Coordinador_BuscaTransfer>');
```

```
SoapSerialize.EndBody;
SoapSerialize.EndEnvelope;
SoapHttpConn.EndMessage;
IF  SoapHttpConn.Error <> '' THEN
BEGIN
  CLEAR(SoapHttpConn);
  ERROR(Text0013);
END;
IF ISCLEAR(XMLDom) THEN
   CREATE(XMLDom);
XMLDom.async := FALSE;
XMLDom.load(SoapHttpConn.OutputStream);
XMLDom.save(cduILR.DameDirXML() + 'transfCoord.xml');
DOMNodelist:= XMLDom.getElementsByTagName('TransferShipmentLineDocumentNo');
IF DOMNodelist.length = 0 THEN
   EXIT(0)
ELSE
    Cuantos := DOMNodelist.length;
txtAux := DOMNodelist.item(i).text;

rcdCabTransfer.RESET;
rcdCabTransfer.SETRANGE("Nº",txtAux);

IF NOT rcdCabTransfer.FINDFIRST() THEN
BEGIN
  rcdCabTransfer.INIT;
  rcdCabTransfer.Tipo:=rcdCabTransfer.Tipo::"Transferencia Coordinador";
  rcdCabTransfer."Nº":=txtAux;
  rcdCabTransfer."Almacen Origen":=Coordinador;
  rcdCabTransfer.Destino:= Conf."Cód. almacén";
  rcdCabTransfer.INSERT;
END;

rcdLinTransfer.RESET;
rcdLinTransfer.SETRANGE(rcdLinTransfer."Nº documento",txtAux);
IF rcdLinTransfer.FINDFIRST THEN
   rcdLinTransfer.DELETEALL();


i:=0;
REPEAT
    rcdLinTransfer.INIT;
    rcdLinTransfer."Nº documento":=rcdCabTransfer."Nº";
    DOMNodelist:= XMLDom.getElementsByTagName('TransferShipmentLineLineNo');
    IF DOMNodelist.length > 0 THEN
      txtAux:=DOMNodelist.item(i).text;
    EVALUATE(rcdLinTransfer."Nº línea",txtAux);
    DOMNodelist:= XMLDom.getElementsByTagName('TransferShipmentLineItemNo');
    IF DOMNodelist.length > 0 THEN
      txtAux:=DOMNodelist.item(i).text;
    rcdLinTransfer."Nº":=txtAux;
    DOMNodelist:= XMLDom.getElementsByTagName('TransferShipmentLineQuantity');
    IF DOMNodelist.length > 0 THEN
      txtAux:=DOMNodelist.item(i).text;
```

```
    EVALUATE(rcdLinTransfer.Cantidad,txtAux);
    DOMNodelist:= XMLDom.getElementsByTagName('TransferShipmentLineDescription');
    IF DOMNodelist.length > 0 THEN
      txtAux:=DOMNodelist.item(i).text;
    rcdLinTransfer.Descripción:=txtAux;
    DOMNodelist:=
XMLDom.getElementsByTagName('TransferShipmentLineCantidadConsumida');
    IF DOMNodelist.length > 0 THEN
      txtAux:=DOMNodelist.item(i).text;
    EVALUATE(rcdLinTransfer."Cantidad enviada",txtAux);

    rcdLinTransfer."Almacen Origen":=rcdCabTransfer."Almacen Origen";
    rcdLinTransfer."Almacen Destino":=rcdCabTransfer.Destino;
    rcdLinTransfer.INSERT;
    i:=i+1;
UNTIL i= Cuantos;
EXIT(Cuantos);
```

TPV_Coordinador_NumTransfer(NumeroTicket)

Esta función se invoca cuando se quiere registrar una entrega de coordinador en una tienda. El sistema busca en CENTRAL si el coordinador ha hecho entregas anteriores de esa transferencia. En caso afirmativo, devuelve la última entrega, para saber con que número se debe hacer el registro. En caso negativo, al número se añade lo siguiente: /001

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si EncabezadoWeb.Vacio entonces
    Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/
                    TPV_Coordinador_NumTransfer');
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXml('<TPV_Coordinador_NumTransfer xmlns=
                http://tempuri.org/">'+'<Usuario>'+ConfigTpv.CodTienda+
                '</Usuario>'+'<password>'+ConfigTpv.Password+'</password>'+
                '<NumeroTicket>'+NumeroTicket+'</NumeroTicket>'+
```

```
                ‘<TPV_Coordinador_NumTransfer>’)
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>’’ entonces
   Limpiar(EncabezadoWeb)
   ERROR(‘Error al conectar con CENTRAL’)
Si EstaVacío(NodoXml) entonces
   Crear(NodoXml)
NodoXml.Cargar(EncabezadoWeb)
NodoXml.Guardar(‘numtransfCoord.xml’)
NodoLista=NodoXml.CogerElementos(TransferEnvioNoCab)
Si NodoLista.longitud=0 entonces
   Salir(NumeroTicket+’/001’)
Sino
   Cuantos=NodoLista.longitud
txtAux=NodoLista.elemento(Cuantos-1).texto
txtAux=Incrementar(txtAux)
Salir(txtAux)
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF NOT Conf.FINDFIRST() THEN
  ERROR('No Existe Configuración TPV');
IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
  URLWS:=Conf."URL Servicio WEB CENTRAL"
ELSE
  ERROR('No existe Configuración del Servicio WEB');



IF ISCLEAR(SoapHttpConn) THEN
  CREATE(SoapHttpConn);

SoapHttpConn.Property('EndPointURL', URLWS);
SoapHttpConn.Connect;

SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_Coordinador_NumTransfer');
SoapHttpConn.BeginMessage;
IF ISCLEAR(SoapSerialize) THEN
  CREATE(SoapSerialize);
SoapSerialize.Init(SoapHttpConn.InputStream);
SoapSerialize.StartEnvelope('','STANDARD','');
SoapSerialize.StartBody('');
SoapSerialize.WriteXML('<TPV_Coordinador_NumTransfer xmlns="http://tempuri.org/">'+
                  '<Usuario>'+Conf."Cód. tienda"+'</Usuario>'+
                  '<password>'+Conf.Password+'</password>'+
                  '<NumeroTicket>'+NumeroTicket +'</NumeroTicket>'+
                  '</TPV_Coordinador_NumTransfer>');
SoapSerialize.EndBody;
SoapSerialize.EndEnvelope;
SoapHttpConn.EndMessage;
IF  SoapHttpConn.Error <> '' THEN
```

```
BEGIN
  CLEAR(SoapHttpConn);
  ERROR(Text0013);
END;
IF ISCLEAR(XMLDom) THEN
   CREATE(XMLDom);
XMLDom.async := FALSE;
XMLDom.load(SoapHttpConn.OutputStream);
XMLDom.save(cduILR.DameDirXML() + 'numtransfCoord.xml');
DOMNodelist:= XMLDom.getElementsByTagName('TransferShipmentHeaderNo');
IF DOMNodelist.length = 0 THEN
   EXIT(NumeroTicket+'/001')
ELSE
    Cuantos := DOMNodelist.length;
txtAux := DOMNodelist.item(Cuantos-1).text;
txtAux:=INCSTR(txtAux);
EXIT(txtAux);
```

TPV_Entrega_Coordinador(Usuario,Password,Ticket)

Esta función se invoca para que cuando se registra una entrega de coordinador en una tienda, los movimientos de producto originados se actualicen en CENTRAL. En este caso, se carga el stock en la tienda y se descuenta del almacén asociado al coordinador.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
       UrlWs:= ConfigTpv.URLCentral
   Sino
       ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si ConfigTpv.ModoTrabajo=OffLine entonces
    Salir(Falso)
Si No ST_Abrir_Login(ConfigTpv.CodTienda,ConfigTpv.Password,Verdadero)
   Salir(Falso)
CabRep se inicializa
CabRep se filtra No sea Ticket
Si no encuentra registros CabRep entonces
    ERROR('Error no encuentro la cabecera de reposicion')
FechaRegistro=Formateo(CabRep.FechaRegistro)
Si EstaVacío(NodoXml) entonces
   Crear(NodoXml)
CadenaInicial='<NewDataSet>'
CadenaFinal='</NewDataSet>'
LineaRep se inicializa
LineaRep se filtra NoDocumento sea Ticket
Si encuentra registros LineaRep entonces
   i=1
   Repetir
```

```
        Cadena(i)=Cadena(i)+'<Lineas>'
        Cadena(i)=Cadena(i)+'<NumLinea>'+Formateo(LineaRep.NoLinea)+
        '</NumLinea>'
        Cadena(i)=Cadena(i)+'<No.>'+LineaRep.No+'</No.>'
        Cadena(i)=Cadena(i)+'<Quantity>'+Formateo(LineaRep.Cantidad)+
        '</Quantity>'
       Cadena(i)=Cadena(i)+'</Lineas>'
         i=i+1
    Hasta siguiente registro LineaRep sea vacío
Si EstaVacío(EncabezadoWeb) entonces
    Crear(EncabezadoWeb)
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/TPV_Entrega_Coordinador);
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
     Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXml('<TPV_Entrega_Coordinador xmlns=http://tempuri.org/>'+
                  '<Usuario>'+Usuario+'</Usuario>'+'<password>'+Password+
                  '</password>'+'<NumeroTicket>'+Ticket+'</NumeroTicket>'+
                  '<AlmOrigen>'+CabRep.AlmacenOrigen+'</AlmOrigen>'+
                 '<AlmDestino>'+CabRep.Destino+'</AlmDestino>'+ '<FechaReg>'+
                 Formateo(CabRep.FechaRegistro)+'</FechaReg>' +'<DatosLineasStr>')
ConectorWeb.EscribirCadena(CadenaInicial)
x=1
Repetir
     ConectorWeb.EscribirCadena(Cadena(i))
     x=x+1
Hasta x>i
ConectorWeb.EscribirCadena(CadenaFinal)
ConectorWeb.EscribirXml('</DatosLineasStr></TPV_Entrega_Coordinador>');
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>'' entonces
        Salir(Falso)
Si EstaVacío(NodoXml)
     Crear(NodoXml)
NodoXml.Cargar(EncabezadoWeb)
NodoXml.Guardar('Entrega_Coord.xml')
Si ComprobarERROR(NodoXml,TxtError) entonces
    ConfigTpv.NoFallos=ConfigTpv.NoFallos+1
    Modificar registro
    Si ConfigTpv.NoFallos>3 entonces
       ConfigTpv.FechaOffLine=Hoy
       ConfigTpv.HoraOffLine=Hoy
       ConfigTpv.ModoTrabajo=OffLine
       Modificar registro
  Salir(Falso)
ConfigTpv.NoFallos=0
Salir(Verdadero)
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF Conf.FINDFIRST THEN
  BEGIN
   IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
     URLWS:=Conf."URL Servicio WEB CENTRAL"
   ELSE
     ERROR('No existe Configuración del Servicio WEB');
  END
ELSE
  ERROR('No Existe Configuración TPV');

IF Conf."Modo de Trabajo" = Conf."Modo de Trabajo"::"Off-Line" THEN
  EXIT(FALSE);
IF NOT ST_Abrir_Login(Conf."Cód. tienda",Conf.Password,TRUE) THEN
  EXIT(FALSE);

CabRep.RESET;
CabRep.SETRANGE("Nº", Ticket);
IF NOT CabRep.FINDFIRST() THEN
  ERROR('ERROR no encuentro la cabecera de reposición registrada.',Ticket);
FechaRegistro      := FORMAT(CabRep."Fecha registro");

IF ISCLEAR(XMLDom) THEN
  CREATE (XMLDom);

CadenaInicial:='<NewDataSet>';
CadenaFinal:='</NewDataSet>';

LinRep.RESET;
LinRep.SETRANGE(LinRep."Nº documento",Ticket);
IF LinRep.FINDSET() THEN
BEGIN
 i:=1;
 REPEAT
   Cadena[i] := Cadena[i] + '<Lineas>';
   Cadena[i] := Cadena[i] + '<NumLinea>'+FORMAT(LinRep."Nº línea")+ '</NumLinea>';
   Cadena[i] := Cadena[i] + '<No.>'+LinRep."Nº"+ '</No.>';
   Cadena[i] := Cadena[i] + '<Item>'+FORMAT(LinRep."Nº")+'</Item>';
   Cadena[i] := Cadena[i] + '<Cantidad>'+FORMAT(LinRep.Cantidad)+'</Cantidad>';
   Cadena[i] := Cadena[i] + '</Lineas>';
   i:=i+1;
 UNTIL LinRep.NEXT=0;
END;

IF ISCLEAR(SoapHttpConn) THEN
  CREATE(SoapHttpConn);

SoapHttpConn.Property('EndPointURL', URLWS);
SoapHttpConn.Connect;
SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_Entrega_Coordinador');
SoapHttpConn.BeginMessage;
```

```
IF ISCLEAR(SoapSerialize) THEN
  CREATE(SoapSerialize);
SoapSerialize.Init(SoapHttpConn.InputStream);
SoapSerialize.StartEnvelope('','STANDARD','');
SoapSerialize.StartBody('');
SoapSerialize.WriteXML('<TPV_Entrega_Coordinador xmlns="http://tempuri.org/">'+
                '<Usuario>'+Usuario+'</Usuario>'+
                '<password>'+Password+'</password>'+
                '<NumeroTicket>'+Ticket+'</NumeroTicket>'+
                '<AlmOrigen>' + CabRep."Almacen Origen" + '</AlmOrigen>' +
                '<AlmDestino>' + CabRep.Destino + '</AlmDestino>' +
                '<FechaReg>' + FORMAT(CabRep."Fecha registro") + '</FechaReg>' +
                '<DatosLineasStr>');

SoapSerialize.WriteString(CadenaInicial);
x:=1;
REPEAT
  SoapSerialize.WriteString(Cadena[x]);
  x:=x+1;
UNTIL x > i;
SoapSerialize.WriteString(CadenaFinal);
SoapSerialize.WriteXML('</DatosLineasStr></TPV_Entrega_Coordinador>');
SoapSerialize.EndBody;
SoapSerialize.EndEnvelope;
SoapHttpConn.EndMessage;
IF SoapHttpConn.Error <> '' THEN
  EXIT(FALSE);
IF ISCLEAR(XMLDom) THEN
  CREATE(XMLDom);

XMLDom.async := FALSE;
XMLDom.load(SoapHttpConn.OutputStream);
XMLDom.save(cduILR.DameDirXML() + 'Entrega_Coord.xml');
IF ComprobarERROR(XMLDom,TxtError) THEN
BEGIN
  Conf."Nº de fallos" +=1;
  Conf.MODIFY;
  IF Conf."Nº de fallos" > 3 THEN
  BEGIN
    Conf."Fecha entrada Off-Line":=WORKDATE;
    Conf."Hora Entrada Off-Line":=TIME;
    Conf."Modo de Trabajo":=Conf."Modo de Trabajo"::"Off-Line";
    Conf.MODIFY;
  END;
  EXIT(FALSE);
END;
Conf."Nº de fallos" :=0;
Conf.MODIFY;
EXIT(TRUE);
```

TPV_Sincronizar_Transportistas(Usuario,Password)

Esta función se invoca cuando la tienda hace la apertura del día, para poder actualizar todos los transportes de los que dispone la empresa para mover mercancía de un lado a

otro. Entonces, para cada movimiento de mercancía se indica el transportista que lo ha realizado, por si ocurre algún problema con la entrega saber a quién se debe reclamar dicho fallo.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si EncabezadoWeb.Vacio entonces
    Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/
                    TPV_Sincronizar_Transportistas');
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXml('<TPV_Sincronizar_Transportistas
                 xmlns=http://tempuri.org/>' + '<Usuario>'+Usuario+'</Usuario>'+
                '<password>'+password+'</password>'+
               '<TPV_Sincronizar_Transportistas>')
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>'' entonces
    Limpiar(EncabezadoWeb)
    ERROR('ERROR al conectar con Central')
Si EstaVacío(NodoXml) entonces
   Crear(NodoXml)
NodoXml.Cargar(EncabezadoWeb)
NodoXml.Guardar('Transportistas.xml')
NodoLista=NodoXml.CogerElementos(TransportistasCodigo)
Si NodoLista.Longitud=0 entonces
    Salir(0)
TablaTransportistas se inicializa
TablaTransportistas se borran todos los registros
i=0
Repetir
   NodoLista=NodoXml.CogerElementos(TransportistasCodigo)
   mCod=NodoLista.elemento(i).texto
   TablaTransportistas se inicializa
   NodoLista=NodoXml.CogerElementos(TransportistasCodigo)
   TablaTransportistas.Codigo=NodoLista.elemento(i).texto
   NodoLista=NodoXml.CogerElementos(TransportistasNombre)
```

```
   TablaTransportistas.Nombre=NodoLista.elemento(i).texto
   Insertar registro TablaTransportistas
   i=i+1
Hasta i=NodoLista.Longitud
Salir(NodoLista.Longitud)
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF Conf.FINDFIRST() THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
   END
ELSE
  ERROR('No Existe Configuración TPV');

IF ISCLEAR(SoapHttpConn) THEN
 CREATE(SoapHttpConn);

SoapHttpConn.Property('EndPointURL', URLWS);
SoapHttpConn.Connect;
SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_Sincronizar_Transportistas');
SoapHttpConn.BeginMessage;
IF ISCLEAR(SoapSerialize) THEN
  CREATE(SoapSerialize);

SoapSerialize.Init(SoapHttpConn.InputStream);
SoapSerialize.StartEnvelope('','STANDARD','');
SoapSerialize.StartBody('');
SoapSerialize.WriteXML('<TPV_Sincronizar_Transportistas xmlns="http://tempuri.org/">'+
                       '<Usuario>'+Usuario+'</Usuario>'+
                       '<password>'+Password+'</password>'+
                       '</TPV_Sincronizar_Transportistas>');
SoapSerialize.EndBody;
SoapSerialize.EndEnvelope;
SoapHttpConn.EndMessage;

IF  SoapHttpConn.Error <> '' THEN BEGIN
   CLEAR(SoapHttpConn);
   ERROR(Text0006);
END;
IF ISCLEAR(XMLDom) THEN
  CREATE(XMLDom);
XMLDom.async := FALSE;
XMLDom.load(SoapHttpConn.OutputStream);
XMLDom.save(cduILR.DameDirXML() + 'Transportistas.xml');

DOMNodelist:= XMLDom.getElementsByTagName('ShippingAgentCode');
IF DOMNodelist.length = 0 THEN
    EXIT(0);
```

```
Transportistas.RESET;
Transportistas.DELETEALL();
i:=0;
REPEAT
  DOMNodelist:= XMLDom.getElementsByTagName('ShippingAgentCode');
  mCod:=DOMNodelist.item(i).text;
  Transportistas.INIT;
  DOMNodelist:= XMLDom.getElementsByTagName('ShippingAgentCode');
  Transportistas.Code:=DOMNodelist.item(i).text;
  DOMNodelist:= XMLDom.getElementsByTagName('ShippingAgentName');
  Transportistas.Name:=DOMNodelist.item(i).text;
  Window.UPDATE(1,ROUND(i / DOMNodelist.length * 10000,1));
  Transportistas.INSERT;
  i:=i+1;
UNTIL i= DOMNodelist.length;
Window.CLOSE;
EXIT(DOMNodelist.length);
```

TPV_Autorizacion_Exceso(Autorizador,Clave,Nombre,FechaCaducidad)

Esta función se invoca cuando en la transferencia existe una línea que se devuelve por exceso. En este caso, salta una pantalla donde se indica quien le autorizó a devolver eso y la clave que ha asignado a la autorización. Entonces, el sistema comprueba si esos datos son válidos y así permitir registrar la transferencia.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si Busca_Autorizacion_Exceso(ConfigTpv.CodAlmacen,Autorizador,Clave,Nombre
   ,FechaCaducidad) entonces
   Salir(Verdadero)
Si Busca_Autorizacion_Exceso('',Autorizador,Clave,Nombre
   ,FechaCaducidad) entonces
   Salir(Verdadero)
Salir(Falso)
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF Conf.FINDFIRST THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
  END
```

```
ELSE
   ERROR('No Existe Configuración TPV');

IF                                                    Busca_Autorizacion_Exceso(Conf."Cód.
almacén",PARAutorizador,PARClaveAutoriza,PARNombre,PARFechaCaduc) THEN
  EXIT(TRUE);
IF
Busca_Autorizacion_Exceso(",PARAutorizador,PARClaveAutoriza,PARNombre,PARFechaCa
duc) THEN
  EXIT(TRUE);
EXIT(FALSE);
```

Busca_Autorizacion_Exceso(Tienda,Autorizador,Clave,Nombre,FechaCaducidad)

Esta función devuelve si existe una autorización con los datos que se han pasado por parámetro. En este caso, existen dos tipos de autorizaciones: Una misma para todas las tiendas o una específica para una tienda. Ese dato viene definido en el parámetro Tienda, es decir, si viene vacío significa que busque una autorización para todas las tiendas o si viene rellena que busque para una tienda en concreto.

PseudoCódigo

```
ConfigTpv se inicializa
ConfigTpv se filtra CodTpv sea NTpv
Si encuentra registro ConfigTpv entonces
   Si ConfigTpv.URLCentral< >'' entonces
      UrlWs:= ConfigTpv.URLCentral
   Sino
      ERROR('No existe Configuración del servicio Web');
Sino
   ERROR('No existe Configuración TPV');
Si EncabezadoWeb.Vacio entonces
    Crear EncabezadoWeb
EncabezadoWeb.Propiedad('URL',UrlWs);
EncabezadoWeb.Conectar;
EncabezadoWeb.Propiedad('Accion','http://tempuri.org/ TPV_Autorizacion_Exceso);
Encabezado.Web.EmpezarMensaje
Si ConectorWeb.Vacio entonces
    Crear ConectorWeb
ConectorWeb.Inicializar;
ConectorWeb.InicioEstructura('','STANDARD','')
ConectorWeb.InicioCuerpo('');
ConectorWeb.EscribirXml('<TPV_Autorizacion_Exceso xmlns=http://tempuri.org/>
                 <UserID>'+ConfigTpv.CodTienda+'</UserID>+'<Password>'+
                ConfigTpv.Password+'</Password>'+'<CodTienda>'+Tienda+
                '</CodTienda>'+'<CodAutorizador>'+Autorizador+'</CodAutorizador>'
                +'<ClaveAutorizacion>'+Clave+'</ClaveAutorizacion>'+
                '</TPV_Autorizacion_Exceso>')
ConectorWeb.FinCuerpo
ConectorWeb.FinEstructura
EncabezadoWeb.FinMensaje
Si EncabezadoWeb.Error<>'' entonces
```

```
   Limpiar(EncabezadoWeb)
   ERROR('Error al conectar con CENTRAL')
   Salir(Falso)
Limpiar(NodoXml)
Si EstaVacío(NodoXml) entonces
    Crear(NodoXml)
NodoXml.Cargar(EncabezadoWeb)
NodoXml.Guardar('AutorizaExceso.xml')
NodoLista=NodoXml.CogerElementos(AutorizacionesTiendasDescripcion)
Si NodoLista.longitud=0 entonces
   Salir(Falso)
Nombre=NodoLista.elemento(0).texto
dvFechaCaduc=0D
NodoLista=NodoXml.CogerElementos(AutorizacionesTiendasFechaCaducidad)
Evaluar(dvFechaCaduc,CopiaCadena(NodoLista.elemento(0).texto,9,2)+'/'+
          CopiaCadena(NodoLista.elemento(0).texto,6,2)+'/'+
          CopiaCadena(NodoLista.elemento(0),texto,1,4))
FechaCaducidad=dvFechaCaduc
Salir(Verdadero)
```

Código Original

```
Conf.RESET;
Conf.SETRANGE("Cód. TPV",cduILR.DameTPV());
IF Conf.FINDFIRST THEN
  BEGIN
    IF Conf."URL Servicio WEB CENTRAL" <> '' THEN
      URLWS:=Conf."URL Servicio WEB CENTRAL"
    ELSE
      ERROR('No existe Configuración del Servicio WEB');
   END
ELSE
   ERROR('No Existe Configuración TPV');
CLEAR(SoapHttpConn);
IF ISCLEAR(SoapHttpConn) THEN
  CREATE(SoapHttpConn);
SoapHttpConn.Property('EndPointURL', URLWS);
SoapHttpConn.Connect;
SoapHttpConn.Property('SoapAction','http://tempuri.org/TPV_Autorizacion_Exceso');
SoapHttpConn.BeginMessage;
IF ISCLEAR(SoapSerialize) THEN
  CREATE(SoapSerialize);
SoapSerialize.Init(SoapHttpConn.InputStream);
SoapSerialize.StartEnvelope('','STANDARD','');
SoapSerialize.StartBody('');
SoapSerialize.WriteXML('<TPV_Autorizacion_Exceso
xmlns="http://tempuri.org/"><UserId>'+Conf."Cód. tienda" +'</UserId>'+
                                    '<Password>'+ Conf.Password+'</Password>' +
                                    '<CodTienda>'+PARTienda + '</CodTienda>'+
                                    '<CodAutorizador>'  +  PARAutorizador  +
'</CodAutorizador>'+
```

```
                                    '<ClaveAutorizacion>'      +PARClaveAutoriza      +
'</ClaveAutorizacion>'
                                   +'</TPV_Autorizacion_Exceso>');

SoapSerialize.EndBody;
SoapSerialize.EndEnvelope;
SoapHttpConn.EndMessage;
IF  SoapHttpConn.Error <> '' THEN
BEGIN
   CLEAR(SoapHttpConn);
   ERROR(Text0001);
   EXIT(FALSE);
END;
CLEAR(XMLDom);
IF ISCLEAR(XMLDom) THEN
  CREATE(XMLDom);
XMLDom.async := FALSE;
XMLDom.load(SoapHttpConn.OutputStream);
XMLDom.save(cduILR.DameDirXML() + 'AutorizaExceso.xml');
DOMNodelist:= XMLDom.getElementsByTagName('AutorizacionesTiendasDescripcion');

IF DOMNodelist.length = 0 THEN
  EXIT(FALSE);

PARNombre:=DOMNodelist.item(0).text;
dvFechaCaduc:=0D;
DOMNodelist:=
XMLDom.getElementsByTagName('AutorizacionesTiendasFechaCaducidadAutorización');
EVALUATE(dvFechaCaduc,COPYSTR(DOMNodelist.item(0).text,9,2)+'/'+COPYSTR(DOM
Nodelist.item(0).text,6,2)+'/'+
      COPYSTR(DOMNodelist.item(0).text,1,4));
PARFechaCaducidad:=dvFechaCaduc;
EXIT(TRUE);
```

5 Seguridad y Base de Datos

5.1 Seguridad

En este apartado se explica la seguridad aplicada a Navision, es decir, todo lo referente a contraseñas de usuarios, caducidad de las contraseñas, permisos y roles de los diferentes usuarios y registro de movimientos de los usuarios.

5.1.1 Usuarios

El administrador del programa es el encargado de crear nuevos usuarios en el sistema. En primer lugar, se crea una nueva sesión de usuario en la base de datos del sistema y a continuación, se crea el usuario en Navision. Una vez realizado estos dos pasos, se debe sincronizar el programa para que el usuario tenga relación con la sesión creada en la base de datos y a la hora de registrar no haya ningún problema.

Por defecto, al usuario se le asignan todos los menús disponibles que existen en el programa. Cada vez que se crea un nuevo usuario en el programa, se pide a su responsable que diga a que menús debe acceder el usuario y así poder quitarle del resto de menús y no tenga acceso a información no autorizada, que pueda utilizarla de forma fraudulenta. Todo este tipo de autorizaciones deben quedar constatadas por escrito, para que si en un futuro ocurriera algo se pudiera alegar utilizando ese escrito.

Por lo general, los usuarios deben ser identificativos, es decir, saber en todo momento a quién pertenece cada usuario. Entonces, la norma que se utiliza para la creación de usuarios es la siguiente: Inicial del Nombre seguido del primer apellido y saber en todo momento la actividad que realizan cada uno de ellos, ya que cada registro que hacen queda grabado el usuario que lo realizó.

Los usuarios tienen una vida limitada, ya que el programa te permite indicar a partir de qué fecha queda inutilizado. Entonces, si el usuario intenta entrar en el programa, el sistema comprueba la fecha indicada en ese campo. Si está dentro de ese rango, no permite al usuario acceder al sistema por más que lo intente.

Por otro lado, a pesar de que la norma dice que por intentar entrar al sistema en un número determinado de veces el sistema debería bloquear al usuario, en este caso no existe un número de intentos determinados para acceder al sistema, ya que si el usuario se equivoca en meter la contraseña o el usuario, el sistema lanza un mensaje diciendo

que uno de los campos es incorrecto pero en ningún caso se bloquea la cuenta y es algo que se debería modificar en el sistema.

Periódicamente, se hace revisión de usuarios por dos motivos: si el usuario ha cambiado de funciones después de la última revisión, se le deben añadir los menús que necesite para poder desempeñar sus nuevas funciones o en el caso de que se le hayan disminuido responsabilidades, se le deberán quitar los menús a los que no está autorizado a entrar.

5.1.2 Contraseñas

Cuando se crea un nuevo usuario se debe indicar qué contraseña va asociada. El administrador como es el responsable de la creación de los usuarios , suele utilizar una contraseña unitaria para facilitar la creación. Por norma, el sistema debería obligar al usuario a cambiar la contraseña unitaria que tiene asignada y así que no se le olvide cambiarla pero en este caso, se le debe indicar como primera premisa que cambie la contraseña lo antes posible, para que sólo él sepa la contraseña y nadie pueda utilizarla de forma fraudulenta. Este hecho se debería modificar en el sistema, porque esto puede ser una vía de escape para hacer uso indebido de la información.

La norma dice que las contraseñas son óptimas si tienen fecha de caducidad, para obligar al usuario a cambiarla cada cierto tiempo y evitar que alguien pueda saberla.En este caso, las contraseñas no tienen caducidad por lo que un usuario puede tener la misma contraseña durante el periodo que quiera. Normalmente, el administrador recomienda a los usuarios que cambien la contraseña cada cierto tiempo, por si en algún momento ha tenido que desvelar su contraseña para realizar un registro en su nombre y, por lo tanto, dicha persona conoce su contraseña. Este hecho se debería modificar en el sistema, para que nadie pueda acceder al sistema con el usuario de otra persona.

Las contraseñas nadie puede tener acceso a ellas, ni el propio administrador, debido a que están cifradas y nadie tiene posibilidad de descifrarlas, por lo que la seguridad de las contraseñas es alta. La única posibilidad de conocimiento de la misma es que el usuario tenga como contraseña la unitaria, que en este caso sería muy fácil acceder a la cuenta.

En la norma, se explica que las contraseñas para que sean de alta seguridad deben contener tanto números como letras y tengan una longitud mínima,por lo tanto el sistema debe obligar a que las contraseñas cumplan ese patrón. En este caso, las

contraseñas no están definidas dentro de un patrón, es decir, no se obliga al usuario a que contenga números y letras, lo único que se les recomienda que no tenga mucha relación con información relevante acerca de ellos, para que no sea fácilmente descifrable. Además, el sistema tampoco obliga a que la longitud de la contraseña tenga unos mínimos de caracteres. Este hecho debería existir dentro del programa, para que las contraseñas no se puedan identificar de forma sencilla.

5.1.3 Roles

En Navision, los usuarios están basados en roles, es decir, al usuario se le deben asignar unos roles para que puedan realizar una serie de funciones o puedan acceder a una serie de formularios. La asignación de roles depende básicamente del responsable de área, quien decide a que menús debe acceder el usuario y dentro de esos menús, que funciones debe realizar y que datos puede visualizar y modificar. Esto es debido a que no todos los usuarios pueden visualizar la misma información, ya que hay parte de la misma que es confidencial y sólo pueden acceder una serie de ellos.

Entonces, cuando el usuario quiere acceder a cualquier formulario de los que tenga visible, lo primero que hace el sistema es comprobar si tiene asignado el rol correspondiente. En caso negativo, el sistema emite un mensaje diciendo que no tiene los permisos suficientes para poder acceder.

Para el caso de la funcionalidad, ocurre algo parecido a los formularios, ya que cuando el usuario quiere registrar, modificar datos o crear nuevos registros, el sistema comprueba que tiene asignados los roles correspondientes. En caso de no tenerlos, el sistema emite un mensaje de error indicando el motivo.

Existen distintos tipos de roles: roles de formularios, roles de botones, roles de funciones, roles de campos. Se hace de esta manera para identificar, de manera clara, que función tiene cada uno de ellos y su asignación se pueda realizar lo más factible posible y evitar posibles errores.

Por otro lado, para cualquier modificación que se quiera realizar sobre un campo, el sistema obliga al usuario a que introduzca una contraseña, que normalmente debe ser su DNI, para validar que es un usuario autorizado para realizar ese tipo de cambios y así asegurarse que no se realizan modificaciones no autorizadas.

5.1.4 Registro

Cualquier registro que se haga en el sistema queda reflejado quién ha sido el autor, ya que si ocurre algún problema en el sistema con alguna operación extraña que se haya hecho y se quiere auditar, se puede saber con facilidad quien ha estado inmerso en los registros involucrados en el problema.

Los usuarios saben de que cualquier registro que se realiza en el sistema queda reflejado el autor, por lo que en ningún momento puede pillarle por sorpresa que vea su usuario reflejado en los movimientos y pueda quejarse por ello.

Antes de empezar a realizar cualquier funcionalidad, se le muestran cuáles son sus derechos y obligaciones en la empresa, por lo que está al corriente, de forma legal, de lo que le puede ocurrir en caso de fraude.

5.1.5 Sesiones

En lo referente a sesiones, el sistema permite que cualquier usuario pueda abrir varias sesiones del programa, ya sea en el mismo terminal o en distintos terminales. Pero existe un proceso, en el cual se indica qué usuarios no están autorizados a tener multisesiones y en caso de que lo quieran intentar, el sistema emite un mensaje diciendo que no puede abrir más de una sesión y por más que lo intente, el sistema se lo deniega.

Por lo general, el sistema permite que cualquier usuario pueda estar dentro del programa en todo momento, sin realizarle bloqueo por timeout. Pero existe un proceso, en el cual se indica qué usuarios no tienen permiso para permanecer en el sistema en todo momento. Entonces, si el usuario sobrepasa un tiempo determinado de inactividad, el sistema echa del programa al usuario.

Todos estos procesos permiten optimizar el uso del programa, ya que este tipo de programa trabaja con un número de licencias determinado. Con esto se asegura que un usuario no permanezca durante un largo tiempo sin poder entrar en el programa y pueda realizar sus funciones diarias sin ningún tipo de problema.

5.2 Base de Datos

En este apartado se habla de la configuración de la base de datos desde la fase más primaria, es decir, realizar los diagramas de flujo de datos, donde se muestra todo el trasiego de información que se produce entre los procesos y los distintos almacenes

configurados,hasta la fase más avanzada en la base de datos, que corresponde al diseño, donde se explican las distintas tablas utilizadas y los campos configurados en cada una de ellas. Además, se muestran las relaciones existentes entre las tablas indicando tipo de relación y que campos están involucrados.

5.2.1 Configuración

En este apartado, se explica, con todo detalle, todo el movimiento de información que se produce entre los distintos procesos. Lo que se denomina los diagramas de flujo, que muestran cómo se comunican los procesos y los almacenes, donde se indican las diferentes llamadas realizadas dentro del proceso y el contenido de las mismas.

Para este caso, se empezará con un diagrama más general, donde se mostrará que personas intervienen en el proceso y las diferentes funciones que pueden realizar. A continuación, se mostrarán los principales módulos de la aplicación, donde se indicará a que tablas accede cada uno para realizar la funcionalidad. Por último, se mostrarán, de forma detallada, todas las funcionalidades de cada uno de los módulos existentes.

Por otro lado, se explica, de manera breve, en que consiste cada llamada existente y, en caso de existir, que campos van inmersos. Además, se explica cada uno de los campos que forman parte de cada una de las tablas.

Diagrama de Flujo de Datos

Diagrama de Contexto

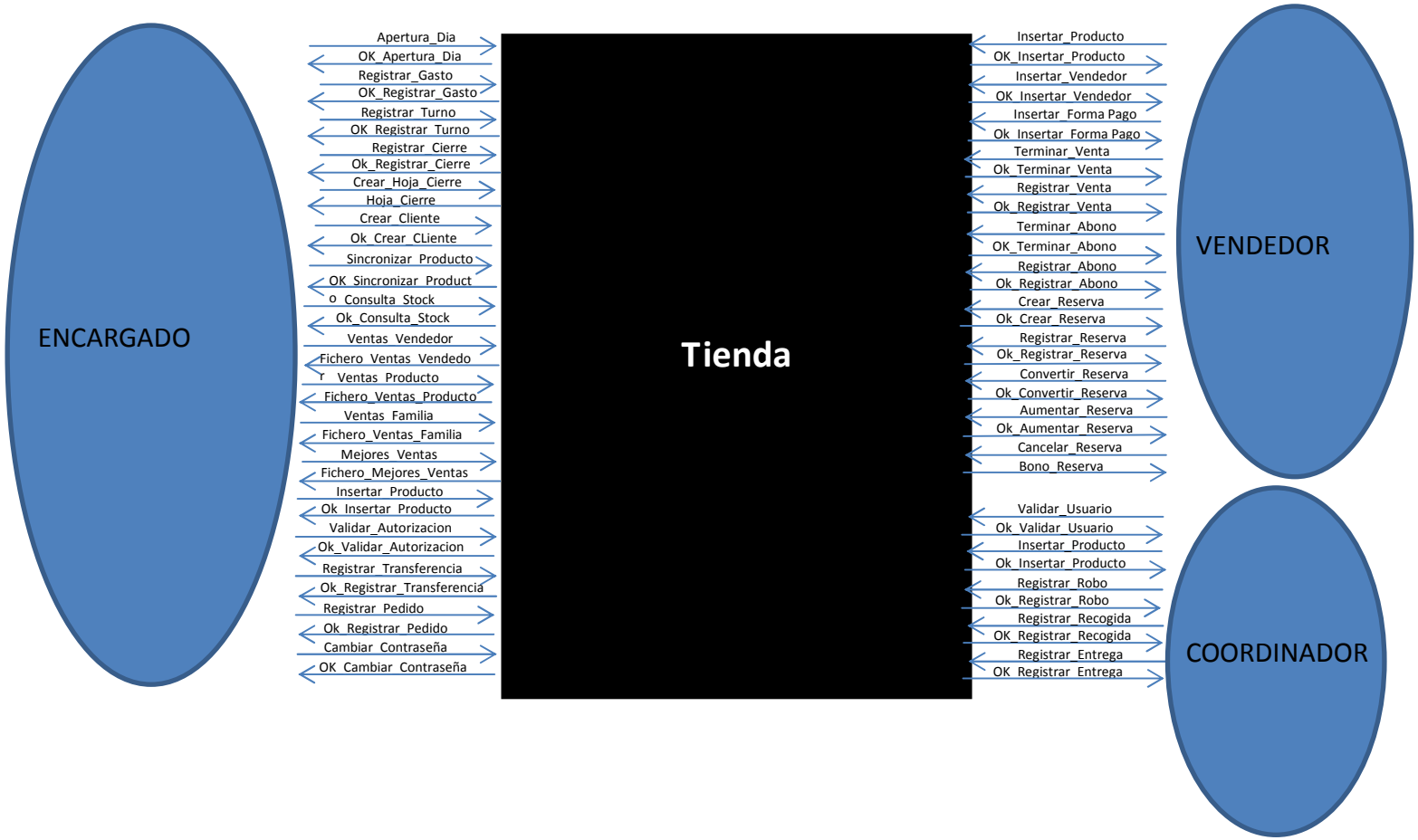
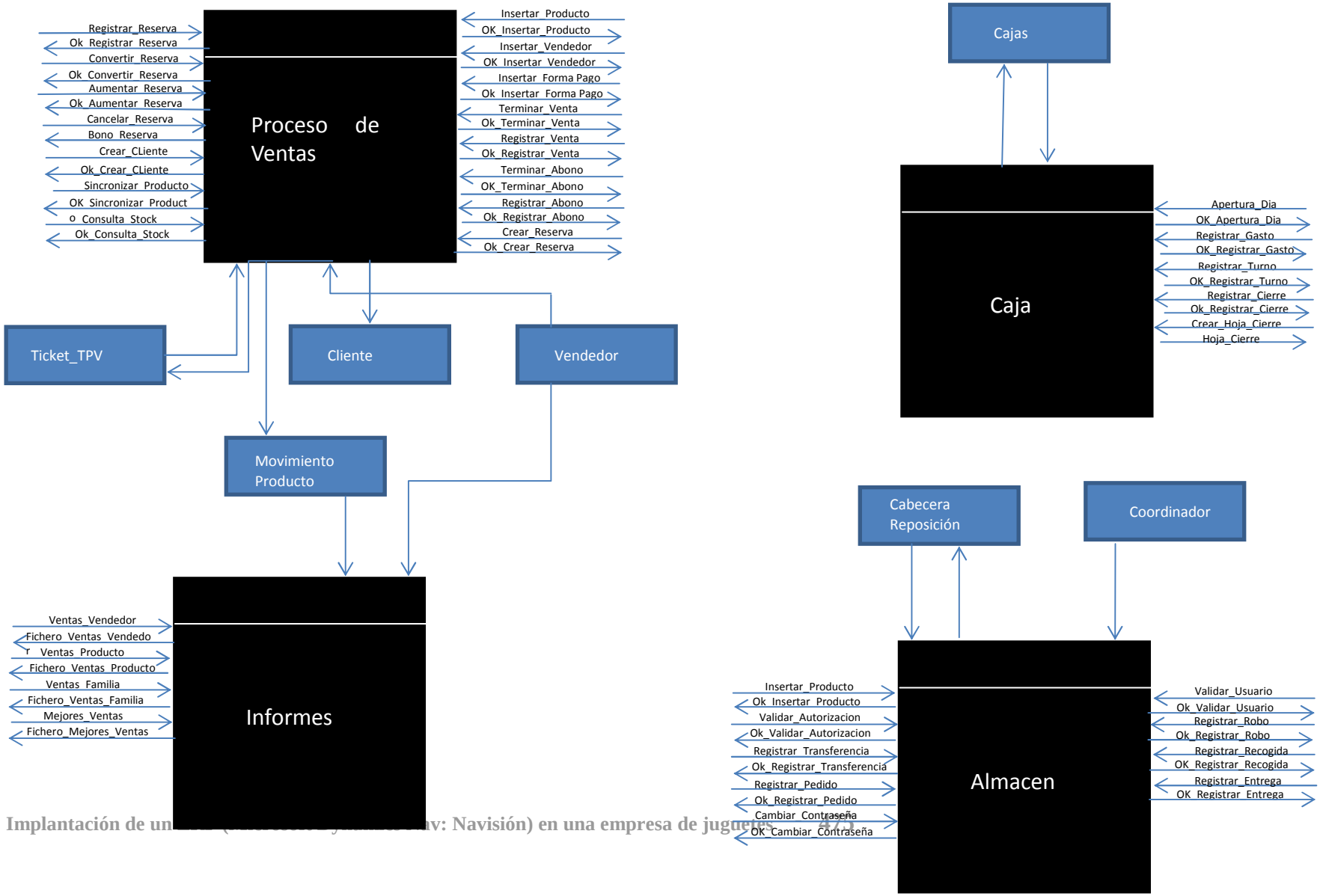
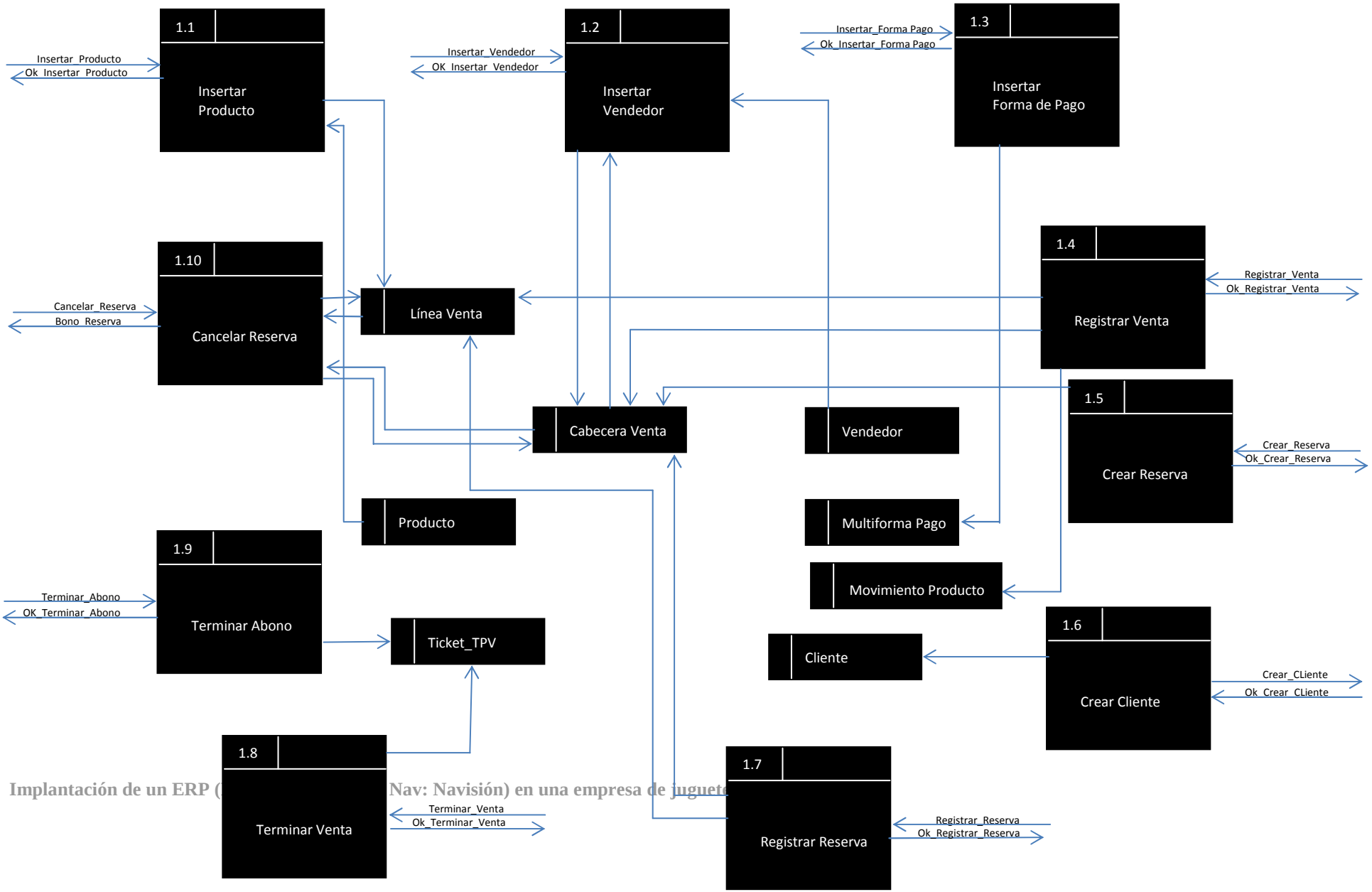

ENCARGADO
Apertura_Dia
OK_Apertura_Dia
Registrar_Gasto
OK_Registrar_Gasto
Registrar_Turno
OK_Registrar_Turno
Registrar_Cierre
Ok_Registrar_Cierre
Crear_Hoja_Cierre
Hoja_Cierre
Crear_Cliente
Ok_Crear_CLiente
Sincronizar_Producto
OK_Sincronizar_Product
o
Consulta_Stock
Ok_Consulta_Stock
Ventas_Vendedor
Fichero_Ventas_Vendedo
r
Ventas_Producto
Fichero_Ventas_Producto
Ventas_Familia
Fichero_Ventas_Familia
Mejores_Ventas
Fichero_Mejores_Ventas
Insertar_Producto
Ok_Insertar_Producto
Validar_Autorizacion
Ok_Validar_Autorizacion
Registrar_Transferencia
Ok_Registrar_Transferencia
Registrar_Pedido
Ok_Registrar_Pedido
Cambiar_Contraseña
OK_Cambiar_Contraseña
Tienda
Insertar_Producto
OK_Insertar_Producto
Insertar_Vendedor
OK_Insertar_Vendedor
Insertar_Forma Pago
Ok_Insertar_Forma Pago
Terminar_Venta
Ok_Terminar_Venta
Registrar_Venta
Ok_Registrar_Venta
Terminar_Abono
OK_Terminar_Abono
Registrar_Abono
Ok_Registrar_Abono
Crear_Reserva
Ok_Crear_Reserva
Registrar_Reserva
Ok_Registrar_Reserva
Convertir_Reserva
Ok_Convertir_Reserva
Aumentar_Reserva
Ok_Aumentar_Reserva
Cancelar_Reserva
Bono_Reserva
Validar_Usuario
Ok_Validar_Usuario
Insertar_Producto
Ok_Insertar_Producto
Registrar_Robo
Ok_Registrar_Robo
Registrar_Recogida
OK_Registrar_Recogida
Registrar_Entrega
OK_Registrar_Entrega
VENDEDOR
COORDINADOR

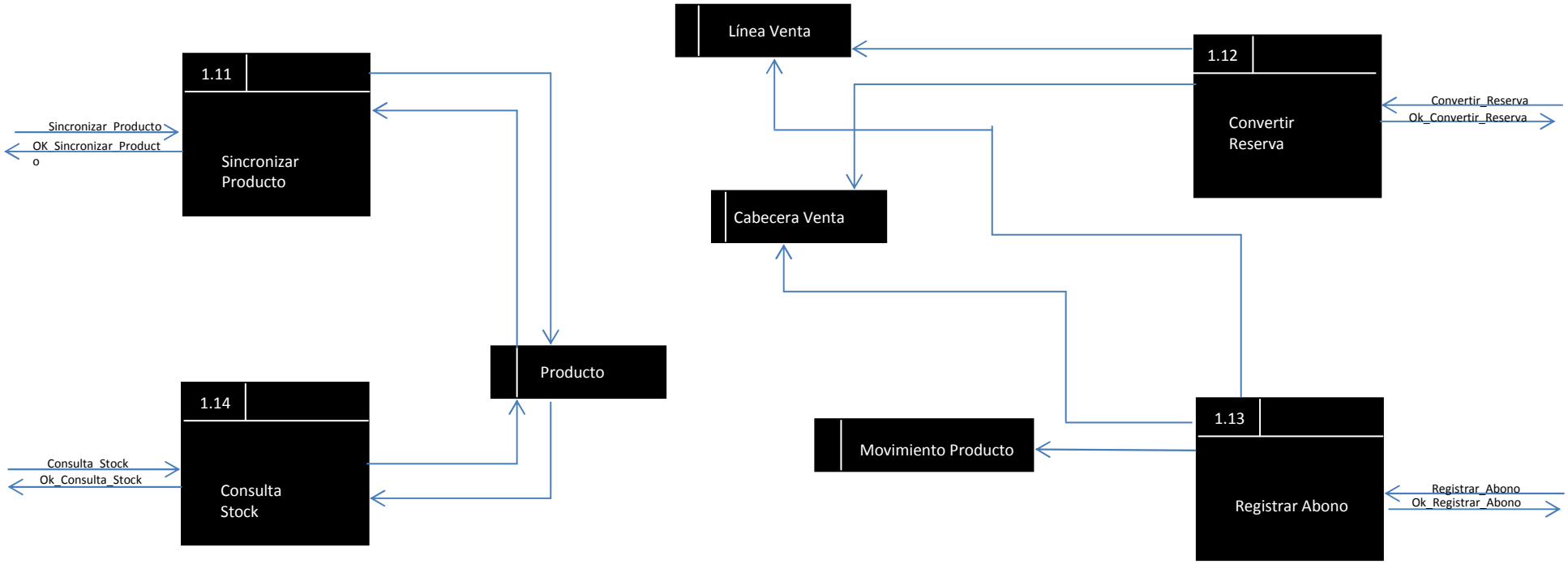
Diagrama de Sistema



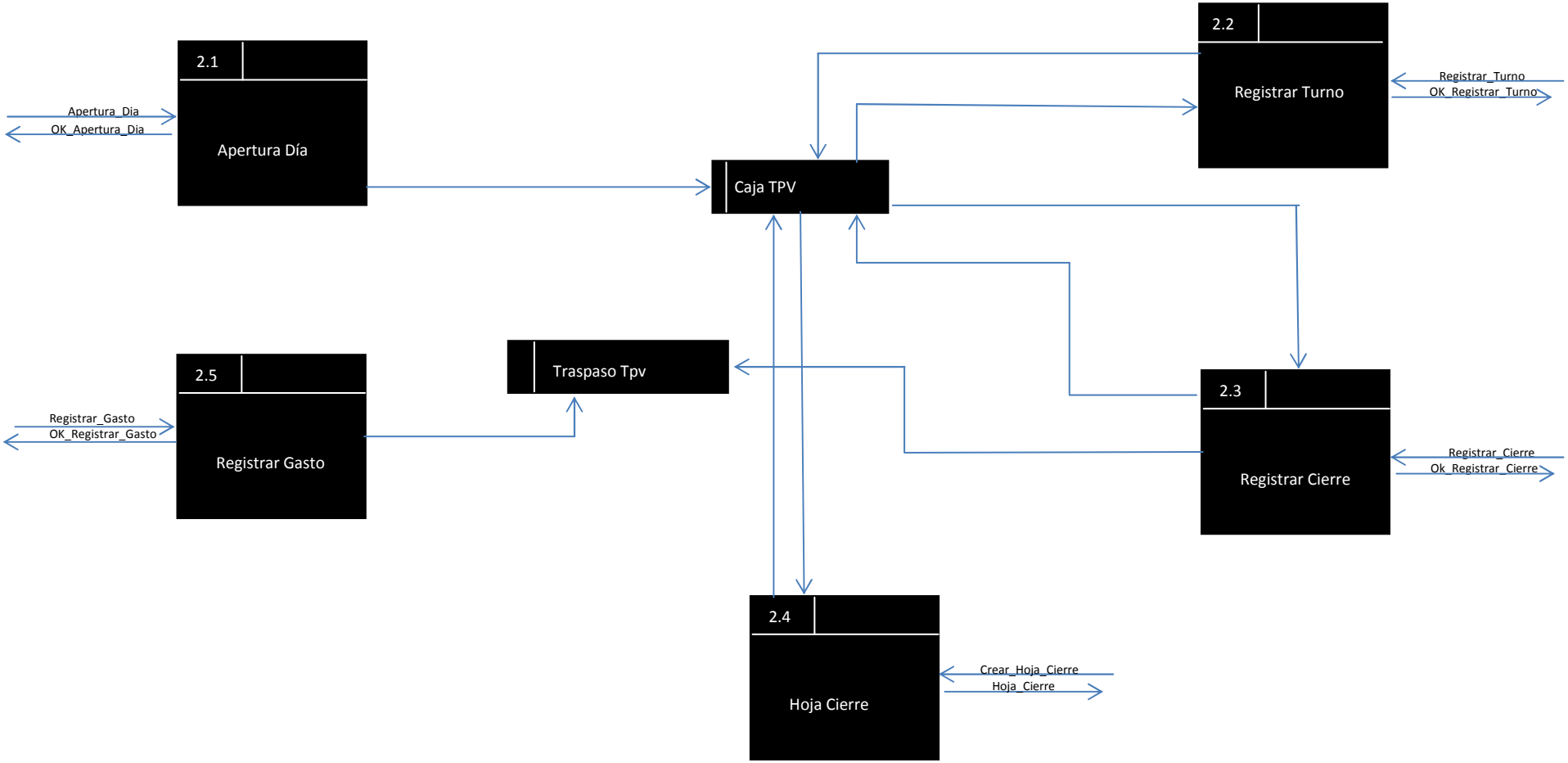
Proceso de Ventas(I)



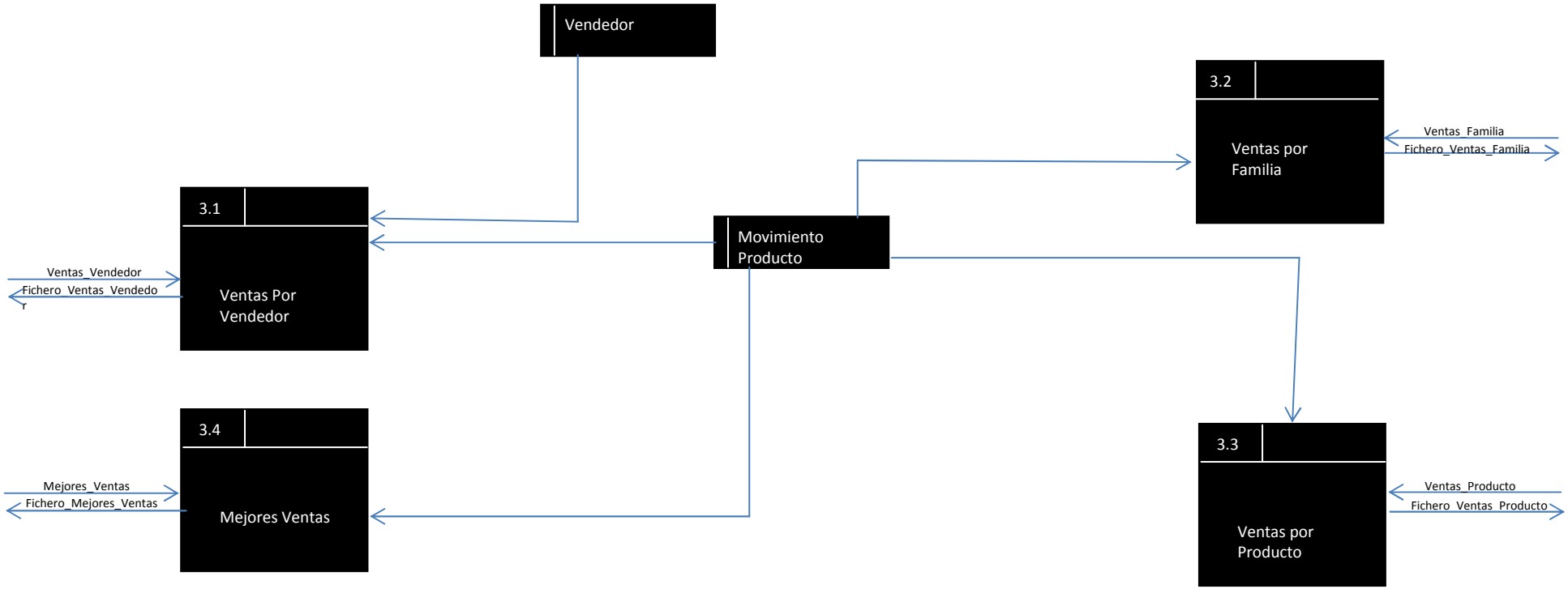
Proceso de Ventas(II)



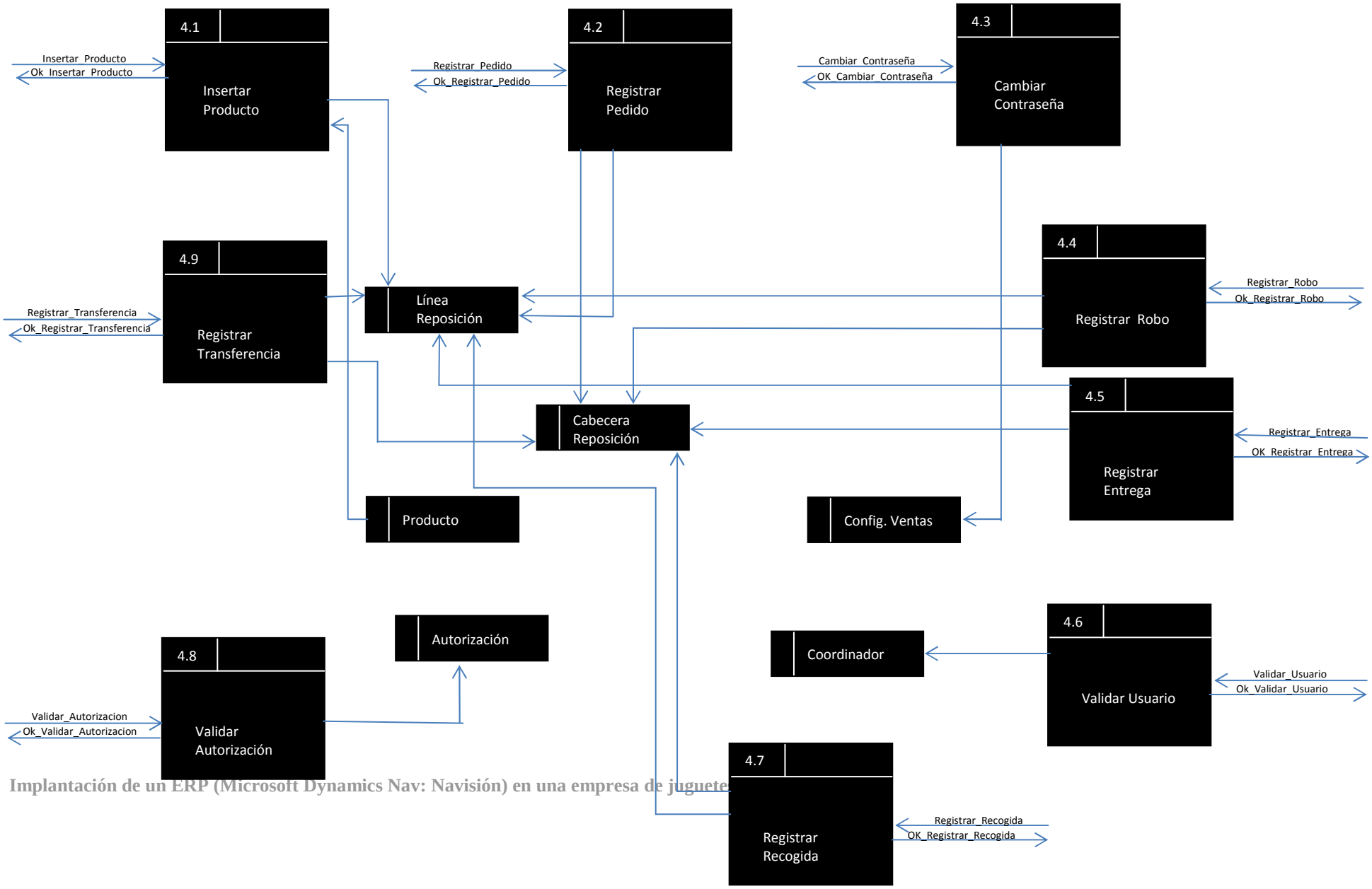
Caja


2.1
Apertura Día
Apertura_Dia
OK_Apertura_Dia
2.2
Registrar Turno
Registrar_Turno
OK_Registrar_Turno
Caja TPV
Traspaso Tpv
2.5
Registrar Gasto
Registrar_Gasto
OK_Registrar_Gasto
2.3
Registrar Cierre
Registrar_Cierre
Ok_Registrar_Cierre
2.4
Hoja Cierre
Crear_Hoja_Cierre
Hoja_Cierre

Informes


Vendedor
3.2
Ventas por Familia
Ventas_Familia
Fichero_Ventas_Familia
3.1
Ventas Por Vendedor
Ventas_Vendedor
Fichero_Ventas_Vendedor
Movimiento Producto
3.4
Mejores Ventas
Mejores_Ventas
Fichero_Mejores_Ventas
3.3
Ventas por Producto
Ventas_Producto
Fichero_Ventas_Producto

Almacén


4.1
Insertar Producto
Insertar_Producto
Ok_Insertar_Producto
4.2
Registrar Pedido
Registrar_Pedido
Ok_Registrar_Pedido
4.3
Cambiar Contraseña
Cambiar_Contraseña
OK_Cambiar_Contraseña
4.9
Registrar Transferencia
Registrar_Transferencia
Ok_Registrar_Transferencia
Línea Reposición
4.4
Registrar Robo
Registrar_Robo
Ok_Registrar_Robo
Cabecera Reposición
4.5
Registrar Entrega
Registrar_Entrega
OK_Registrar_Entrega
Producto
Config. Ventas
Autorización
4.8
Validar Autorización
Validar_Autorizacion
Ok_Validar_Autorizacion
Coordinador
4.6
Validar Usuario
Validar_Usuario
Ok_Validar_Usuario
4.7
Registrar Recogida
Registrar_Recogida
OK_Registrar_Recogida

Definición de los Flujos de Datos

En este apartado, se explica, de manera breve, en que consiste cada uno de los flujos de datos que forman parte de la funcionalidad del sistema.

- **Apertura_Día**: Contiene los datos para registrar la apertura del día. CodTienda, CodTpv, FechaInicio, HoraInicio, Saldo Inicial, ErrorEnvio
- **Ok_Apertura_Día**: Confirma que el registro de la apertura se ha hecho de forma correcta. Ok o Código Error, FechaInicio, CodTpv
- **Registrar Gasto**: Contiene los datos para registrar un gasto que haya realizado la tienda. Nº Documento, Tipo Gasto, Descripcion, Importe, Cuenta Origen, Fecha
- **Ok_Registrar_Gasto**: Confirma que el registro del gasto se haya hecho de manera satisfactoria. Ok o Código Error, NºDocumento
- **Registrar_Turno**: Contiene los datos para registrar el primer turno del día. CodTienda, CodTpv, FechaFin, HoraFin, DiferenciaCaja, DiferenciaTarjeta, SaldoFinal, ImporteaIngresar
- **Ok_Registrar_Turno**: Confirma que el registro del turno se ha realizado correctamente. OK o Código Error, FechaInicio, CodTpv
- **Registrar_Cierre**: Contiene los datos para registrar el cierre del día. CodTienda, CodTpv, FechaFin, HoraFin, DiferenciaCaja, DiferenciaTarjeta, SaldoFinal, ImporteaIngresar, ImporteTotal
- **Ok_Registrar_Cierre**: Confirma que el registro de cierre de día se ha realizado correctamente. OK o Código Error, FechaInicio, CodTpv
- **Crear_Hoja_Cierre**: Contiene los datos para que el sistema genere un fichero de Excel con todos los datos de la caja del día que haya pasado por parámetro. FechaInicio
- **Hoja_Cierre**: Fichero de Excel que contiene todos los datos relacionados con la caja del día. VentasTarjeta, VentasEfectivo, VentasBono, AbonosTarjeta, AbonosEfectivo, AbonosBono, ReservasTarjeta, ReservasEfectivo, ReservasBono, Gastos, SaldoInicial, RecuentoTarjeta, RecuentoEfectivo
- **Crear_Cliente**: Contiene los datos para dar de alta un nuevo cliente en la base de datos. Nº, CIF, Nombre, Direccion, CP, Provincia, Telefono
- **Ok_Crear_Cliente**: Confirma que la inserción del nuevo cliente en la base de datos se ha hecho de forma correcta. Ok o Código Error, Nº

- **Sincronizar_Producto**: Contiene los datos para sincronizar el producto indicado y tenerlo actualizado al momento. Nº Producto

- **Ok_Sincronizar_Producto**: Confirma que la actualización del producto ha sido satisfactoria. Ok o Código Error, Nº Producto

- **Consulta Stock**: Contiene el número de producto que se quiere consultar el stock existente en cada tienda. Nº Producto

- **Ok_Consulta_Stock**: Devuelve una pantalla con el stock de cada tienda si la consulta ha sido de forma correcta. En caso negativo, devolverá mensaje de error, Nº Producto

- **Ventas_Vendedor**: Contiene los datos para consultar las ventas realizadas por un vendedor en un periodo de tiempo. Nº Vendedor, FechaRegistro

- **Fichero_Ventas_Vendedor**: Devuelve una pantalla dónde se muestra el valor de las ventas realizadas por el vendedor. También lo muestra en Excel

- **Ventas_Producto**: Contiene los datos para consultar las ventas, en unidades, de cada producto en un periodo de tiempo. Nº Producto, FechaRegistro

- **Fichero_Ventas_Producto**: Devuelve una pantalla donde se muestran las unidades vendidas por cada producto en un rango de fechas indicado. También lo muestra en Excel

- **Ventas_Familia**: Contiene los datos para consultar las ventas, en unidades y valor, de cada una de las familias existentes en un periodo de tiempo.
Familia, Fecha Registro

- **Fichero_Ventas_Producto**: Devuelve una pantalla donde se muestran las unidades vendidas y el valor de esas ventas por cada familia. Al final muestra el total de unidades y el valor total. También lo muestra en Excel

- **Mejores_Ventas**: Contiene información de los mejores productos según el criterio que se haya marcado en las opciones. Familia, TipoCriterio, NúmeroCodigosaMostrar

- **Fichero_Mejores_Ventas**: Devuelve una pantalla con los productos que entran dentro del top, donde se muestra Código Producto, Descripción, Ventas Unidades, Ventas valor. Al final muestra ventas totales de esos productos. También se puede mostrar en Excel.

- **Insertar_Producto**: Contiene los datos para insertar un producto en una línea de transferencia. Nº Producto, Cantidad, Tipo Cabecera, Nº Cabecera, Nº Linea

- **Ok_Insertar_Producto**: Confirma que la inserción ha sido correcta. Ok, Código Error, Nº Producto

- **Validar_Autorizacion**: Contiene los datos para comprobar que la autorización para la devolución de los productos es correcta. NombreAutorizador, CodigoAutorizador.

- **Ok_Validar_Autorizacion**: Confirma que la autorización que se ha indicado es la correcta. Ok o Código Error, NombreAutorizador

- **Registrar_Transferencia**: Contiene los datos para registrar la transferencia y en Central puedan manipular la mercancía. Nº Transferencia

- **Ok_Registrar_Transferencia**: Confirma que el registro de la transferencia se ha hecho de forma correcta. Ok o Código Error, Nº Transferencia

- **Registrar_Pedido**: Contiene los datos para registrar el pedido y el almacén pueda prepararlo. Nº Pedido

- **Ok_Registrar_Pedido**: Confirma que el registro del pedido se ha hecho de forma correcta. Ok o Código Error, Nº Pedido

- **Cambiar_Contraseña**: Contiene la información para cambiar la contraseña de cambio de precios, que es necesario hacerlo cada cierto tiempo. ContraseñaAntigua, ContraseñaNueva

- **Ok_Cambiar_Contraseña**: Confirma que el cambio de la contraseña se ha hecho correctamente. OK o Código Error

- **Insertar_Producto**: Contiene los datos para insertar un producto en una línea de venta. Nº Producto, Cantidad, Tipo Venta, Nº Venta, Nº Linea

- **Ok_Insertar_Producto**: Confirma que la inserción ha sido correcta. Ok, Código Error, Nº Producto

- **Insertar_Vendedor**: Contiene los datos para insertar el número de vendedor en la cabecera de la venta. Nº Vendedor

- **Ok_Insertar_Vendedor**: Confirma que la inserción se ha realizado de forma satisfactoria. Ok o Código Error, Nº Vendedor

- **Insertar_Forma Pago**: Contiene los datos para insertar la forma de pago que se va utilizar en la venta. Código Forma Pago, Importe, Devuelto

- **Ok_Insertar_ Forma Pago**: Confirma que la inserción del registro se ha hecho correctamente. Ok o Código Error, Codigo Forma Pago

- **Terminar_Venta**: Contiene los datos para crear un nuevo registro de la tabla Ticket_TPV . Nº Cabecera Venta
- **Ok_Terminar_Venta**: Confirma que la creación del registro se ha hecho correctamente Ok o Código Error, Nº Cabecera Venta
- **Terminar_Abono**: Contiene los datos para crear un nuevo registro de la tabla Ticket. Nº Cabecera Venta
- **Ok_Terminar_Abono**: Confirma que la creación del registro se ha hecho correctamente Ok o Código Error, Nº Cabecera Venta
- **Registrar_Venta**: Contiene los datos para registrar la venta y generar el movimiento de producto correspondiente. Nº Cabecera Venta
- **Ok_Registrar_Venta**: Confirma que el registro de la venta se ha hecho de forma satisfactoria. OK o Código Error, Nº Cabecera Venta
- **Registrar_Abono**: Contiene los datos para registrar el abono y generar el movimiento de producto correspondiente. Nº Cabecera Venta
- **Ok_Registrar_Abono**: Confirma que el registro del abono se ha hecho de forma satisfactoria. OK o Código Error, Nº Cabecera Venta
- **Crear Reserva**: Contiene los datos para crear una nueva reserva en el sistema. Nº Serie Reserva
- **Ok_Crear_Reserva**: Confirma que la inserción de un nuevo registro se ha realizado sin problemas. Ok o Código Error.
- **Registrar_Reserva**: Contiene los datos para registrar la reserva de una serie de productos. Nº Cabecera Venta
- **Ok_Registrar_Reserva**: Confirma que el registro de la reserva se ha hecho de forma satisfactoria. OK o Código Error, Nº Cabecera Venta
- **Convertir_Reserva**: Contiene los datos para generar una venta a partir de la reserva, debido a que el cliente ha ido a recoger los productos que reservó en su momento. Nº Cabecera Venta
- **Ok_ Convertir_Reserva**: Confirma que la creación de la venta se ha hecho de forma correcta. Ok o Código Error, Nº Cabecera Venta
- **Aumentar Reserva**: Contiene los datos para desbloquear la reserva y asi el cliente puede dejar una nueva señal por esos productos. Nº Cabecera Venta
- **Ok_Aumentar_Reserva**: Confirma que el desbloqueo de la reserva se ha hecho de forma satisfactoria. Ok o Código Error, Nº Cabecera Venta

- **Cancelar_Reserva**: Contiene los datos para cancelar la reserva y devolver al cliente el bono que se genera con la cancelación. Nº Cabecera Venta
- **Bono_Reserva**: Contiene la información del bono que se genera con la cancelación de la reserva. Nº Bono, Importe, Importe Liquidado
- **Validar_Usuario**: Contiene la información para comprobar que el usuario, de tipo coordinador, que se ha tecleado es correcto. Codigo Coordinador, Contraseña.
- **Ok_Validar_Usuario**: Confirma si el usuario introducido es correcto. Ok o Código Error, Codigo Coordinador.
- **Insertar_Producto**: Contiene los datos para insertar un producto en una línea de transferencia de tipo coordinador. Nº Producto, Cantidad, Tipo Transferencia, Nº Cabecera Reposicion, Nº Linea
- **Ok_Insertar_Producto**: Confirma que la inserción ha sido correcta. Ok, Código Error, Nº Producto
- **Registrar_Robo**: Contiene los datos para registrar los artículos que han sido robados para que se puedan generar los movimientos de producto correspondientes. Nº Cabecera Reposicion
- **Ok_Registrar_Robo**: Confirma que el registro del robo se ha hecho de forma correcta. Ok o Código Error, Nº Cabecera Reposicion
- **Registrar_Recogida**: Contiene los datos para registrar los productos que se lleva el coordinador de la tienda para repartirlos en otra. Nº Cabecera Reposicion
- **Ok_Registrar_Recogida**: Confirma que el registro de la recogida de productos se ha hecho de forma correcta. Ok o Código Error, Nº Cabecera Reposicion
- **Registrar_Entrega**: Contiene los datos para registrar la entrega de una serie de productos en una tienda por parte del coordinador. Nº Cabecera Reposicion
- **Ok_Registrar_Entrega**: Confirma que el registro de la entrega se ha hecho de forma correcta. Ok o Código Error, Nº Cabecera Reposicion

Definición de los almacenes de Datos

En este apartado se especifican los campos que tiene cada uno de los almacenes.

Ticket_TPV

Contiene todos los movimientos que se han realizado de las ventas, ya sea venta o abono.

Atributo	Descripción	Tipo
Tipo Documento	Tipo de Documento generado	Opción
Nº Documento	Nº Documento generado(ID de la tabla)	Texto
CodTienda	Código de la tienda del ticket	Texto
CodTpv	Código del número de caja correspondiente	Texto
Fecha Documento	Fecha del Ticket	Fecha
Hora Documento	Hora del Ticket	Tiempo
Nº Doc Registrado	Nº Documento asociado a ese ticket	Texto
Total Venta	Importe de la venta	Decimal
Total a Pagar	Importe que debe pagar el cliente	Decimal
Registrado	Si ha sido registrado el documento	Booleano
Total Venta Registrada	Importe de la venta registrada	Decimal
Total Devolución	Importe de la devolución	Decimal
Total Devolución Registrada	Importe de la devolución registrada	Decimal
Importe Entregado	Importe entregado por el cliente	Decimal
Vendedor	Vendedor que ha realizado la venta	Entero Texto
Forma Pago	Forma de pago con la que se ha registrado la venta	Texto

Tipo Pago	Tipo de pago asociado a la forma de pago	Opción
Origen Devolución	Donde proviene la devolución	Opción
CodCliente	Código Cliente al que se le ha generado la venta	Texto
Motivo Devolución	Motivo de la devolución de los productos	Texto

Cliente

Contiene todos los clientes dados de alta en la tienda

Atributo	Descripción	Tipo
Numero	Identificador del Cliente	Texto
CIF	NIF del Cliente	Texto
Nombre	Nombre del Cliente	Texto
Direccion	Direccion del Cliente	Texto
Código Postal	Código Postal del Cliente	Entero
Poblacion	Población del Cliente	Texto
Pais	País del Cliente	Texto
Telefono	Teléfono del cliente	Entero
Email	Email del cliente	Texto
Error Envio	Si se ha enviado el cliente a la base Central	Booleano
Codigo Vendedor	Vendedor que ha dado de alta el cliente	Entero
Fecha Ult.Mod	Fecha en que se hizo la última modificación	Fecha
Alias	Alias del Cliente	Texto

Vendedor

Contiene todos los vendedores dados de alta en el sistema

Atributo	Descripción	Tipo
Codigo	Identificador del vendedor	Entero
Nombre	Nombre del Vendedor	Texto
%Comision	% Comisión sobre la venta	Decimal
Fecha. Ult. Mod	Fecha de la última modificación sobre el vendedor	Fecha

Movimiento Producto

Contiene todos los movimientos de producto que ha ido generando la tienda mediante venta, abono, transferencia, etc.

Atributo	Descripción	Tipo
Numero	Identificador del movimiento de producto	Entero
Nº Producto	Producto asociado al movimiento	Texto
Fecha Registro	Fecha del movimiento	Fecha
Tipo	Tipo de movimiento	Opcion
Cod. Procedencia	Código Procedencia del movimiento	Texto
Nº Documento	Nº Documento asociado al movimiento	Texto
Descripcion	Descripción del Producto	Texto
Almacen	Almacén donde se carga el movimiento	Texto
Cantidad	Cantidad que se mueve en ese movimiento	Entero
Tipo Procedencia	Tipo Procedencia del movimiento	Opción
ImporteVenta	Importe de la Venta	Decimal

Caja_TPV

Contiene todas las aperturas del día hechas en la tienda.

Atributo	Descripción	Tipo
CodTienda	Código de la Tienda	Texto
CodTpv	Caja de la Tienda	Texto
Fecha Inicio	Fecha Inicio del día	Fecha
Hora Inicio	Hora Inicio del día	Tiempo
Fecha Fin	Fecha Fin del día	Fecha
Hora Fin	Hora Fin del día	Tiempo
Saldo Inicial	Saldo Inicial de la Caja	Decimal
Diferencia Caja	Diferencia entre lo contado y lo registrado	Decimal
Fecha Ult. Mod	Fecha Modificación de la Caja	Fecha
Total Recuento	Total recontado de monedas y billetes	Decimal
Total Recuento Tarjeta	Total recontado en Tarjeta	Decimal
Diferencia Tarjeta	Diferencia entre lo registrado y lo recontado	Decimal
Registrado	Si el cierre del día está registrado	Booleano
Activa	Si la caja se encuentra activa	Booleano
Nº Turno	Nº turno correspondiente a la caja	Entero
Apertura	Si se ha hecho la apertura de la caja	Booleano
Saldo final TPV	Saldo final de la caja	Decimal
Total gastos TPV	Total de gastos del día	Decimal
Ventas tarjeta	Ventas realizadas en tarjeta	Decimal
Ventas Efectivo	Ventas realizadas en efectivo	Decimal
Ventas BonoPoly	Ventas realizadas en bono	Decimal

Reservas tarjeta	Reservas realizadas por tarjeta	Decimal
Reservas efectivo	Reservas realizadas por efectivo	Decimal
Reservas BonoPoly	Reservas realizadas por bono	Decimal
Abonos Tarjeta	Abonos realizados en tarjeta	Decimal
Abonos Efectivo	Abonos realizados en efectivo	Decimal
Abonos BonoPoly	Abonos realizados en bono	Decimal
Reserva Cancelada tarjeta	Reservas canceladas por tarjeta	Decimal
Reserva Cancelada Efectivo	Reservas Canceladas en efectivo	Decimal
Reserva Cancelada Bono	Reservas canceladas en bono	Decimal
Error Envio	Si la caja no se ha podido enviar a Central	Booleano
Importe a Ingresar	Importe a ingresar en el banco	Decimal

Cabecera Reposición

Contiene todas las transferencias realizadas en la tienda, ya sea pedido, robo, entrega del coordinador, recogida del coordinador y transferencia.

Atributo	Descripción	Tipo
Nº	Identificador de la Tabla	Texto
Almacén Origen	Almacén de dónde sale	Texto
Descripción Origen	Descripción del Almacén Origen	Texto
Almacén Destino	Almacén donde debe llegar	Texto
Descripción Destino	Descripción del Almacén destino	Texto

Fecha Creacion	Fecha Creación de la Transferencia	Fecha
Fecha Registro	Fecha registro de la Transferencia	Fecha
Usuario	Usuario de creación de la transferencia	Texto
Tipo	Tipo deTransferencia	Opción
Hora Creación	Hora de la creación de la transferencia	Hora
Estado	Estado de la transferencia	Opción

Línea Venta

Contiene cada una de las líneas que componen las facturas que se han ido registrando, ya sea reserva, factura o abono.

Atributo	Descripción	Tipo
Tipo Documento	Tipo de Documento Asociado	Opción
Nº Documento	Identificador de la tabla	Texto
Nº Línea	Nº Línea del Documento	Entero
Cliente	Cliente del Documento	Texto
Tipo	Tipo de Línea	Opción
Nº Producto	Número de Producto	Texto
Almacén	Almacén asociado a la factura	Texto
Fecha Registro	Fecha Registro del documento	Fecha
Descripción	Descripción del Producto	Texto
Cantidad	Cantidad vendida del producto	Entero
%Descuento	% Descuento aplicado al precio	Decimal
Importe	Importe vendido del	

	producto	Decimal

Cabecera Venta

Contiene la cabecera de cada uno de los documentos que se han ido registrando

Atributo	Descripción	Tipo
Nº	Identificador de la tabla	Texto
Tipo Documento	Tipo Documento de la factura	Option
Nº Cliente	Número de Cliente al que se factura	Texto
NombreCliente	Nombre del Cliente al que se factura	Texto
Fecha Registro	Fecha de la facturación	Fecha
Terminos de Pago	Término en que se pagó la factura	Texto
Almacén	Almacén donde se ha imputado la factura	Texto
Grupo Contable Cliente	Grupo Contable al que pertenece el cliente	Texto
Grupo Precio Cliente	Grupo precio al que pertenece el cliente	Texto
CodVendedor	Código del Vendedor	Entero
Grupo Contable Negocio	Grupo Contable al que pertenece la factura	Texto
Importe	Importe de la factura	Decimal

Productos

Contiene todos los productos con los que trabajan en la tienda

Atributo	Descripción	Tipo
Nº	Identificador de la tabla	Texto
Descripción	Descripción del producto	Texto
Subfamilia	Subfamilia que tiene	

	asignada el producto	Entero
Grupo Contable	Grupo Contable del producto	Texto
Precio Venta	Precio de Venta que tiene el producto	Decimal
Código Proveedor	Código que tiene el producto asignado por el proveedor	Texto
Fecha Ult.Mod	Fecha de la última modificación	Fecha
IVA	IVA asignado al producto	Entero

En este caso, la tabla producto tiene conexión con otras dos tablas, que son: Código de Barras y Tarifa Producto

Código Barras

Atributo	Descripción	Tipo
Producto	Identificador de la Tabla	Texto
Código Barras	Código de barras del proveedor asociado al producto	Texto

Precio Venta

Atributo	Descripción	Tipo
Codigo Producto	Identificador de la tabla	Texto
Tipo Venta	A qué tipo de clientes se les aplica dicha tarifa	Opción
Código Venta	Grupo Contable Precio al cual aplicar oferta	Texto
Fecha Inicial	Fecha de cuando entra en vigor la tarifa	Fecha
Fecha Final	Fecha final de cuando termina la tarifa	Fecha

Precio Oferta	Importe de la tarifa	Decimal

Multiforma de Pago

Contiene las distintas formas de pago que se han utilizado para pagar cada una de las facturas existentes

Atributo	Descripción	Tipo
Tipo Documento	Tipo de documento el cual se ha generado la forma de pago	Opción
Nº Documento	Nº Documento asociado a la forma de pago	Texto
Nº Línea	Número de línea correspondiente a esa forma de pago	Entero
Cod.FormaPago	Código de la forma de pago realizada en esa factura	Texto
Tipo Contrapartida	Tipo de contrapartida para esa forma de pago	Opción
Cuenta Contrapartida	Cuenta asignada a esa contrapartida	Entero
Importe	Importe asociado a esa forma de pago	Decimal
Cambio	Indica si se le tiene que devolver dinero al cliente	Decimal
CodTienda	Código de la Tienda	Entero
CodTpv	Código de la Caja	Entero
Fecha registro	Fecha registro de la factura	Fecha
Nº Bono	En caso de que la forma de pago sea bono, indica qué bono se ha utilizado	Texto
Registrado	Indica si esa factura ha sido	

	registrada	Booleano
Datafono	Devuelve la respuesta del datafono virtual	Texto

Trapasos_Tpv

Contiene todos los traspasos de cuenta que se han hecho en la tienda y que pueden tener como origen el cierre del día, donde se realiza el traspaso de lo que la tienda ingresa al día siguiente al banco y lo que deja de saldo inicial para el siguiente día. También se puede deber al registro de un gasto generado por la tienda.

Atributo	Descripción	Tipo
Nº	Identificador de la Tabla	Entero
CodTienda	Código de la Tienda	Entero
CodTpv	Código de la Caja	Entero
FechaInicio	Fecha del movimiento	Fecha
Usuario	Usuario que realizó el movimiento	Texto
Hora	Hora del movimiento	Tiempo
Nº Documento	Número de documento asociado al movimiento	Texto
Importe	Importe del movimiento	Decimal
Cuenta	Cuenta dónde se imputa el movimiento	Entero
Signo	Signo del movimiento	Texto

Línea Reposición

Contiene todas las líneas que se han generado de cada una de las transferencias, ya sea pedido, transferencia de coordinador, etc

Atributo	Descripción	Tipo
Nº Documento	Número de documento dónde se encuentra la línea	Texto
Nº Linea	Número de Línea asociada al documento	Entero

Nº	Código del Producto	Texto
Descripción	Descripción del producto	Texto
Cantidad	Cantidad a enviar	Entero
AlmacenOrigen	Almacén de donde saldrá la mercancía	Texto
AlmacenDestino	Almacén donde llegará la mercancía	Texto
Cantidad Enviada	Cantidad que ha enviado un coordinador en alguna entrega	Entero

Autorización

Contiene todas las autorizaciones creadas para que puedan hacer devolución de mercancía a Central.

Atributo	Descripción	Tipo
Código Autorizador	Nombre Autorizador	Texto
Contraseña Autorizador	Contraseña para la autorización	Texto
Fecha Inicio	Fecha cuando entra en vigor	Fecha
Fecha Final	Fecha cuando deja de tener vigencia	Fecha

Config.Ventas

Contiene la contraseña para cuando se quiere realizar un cambio de precio en un producto.

Atributo	Descripción	Tipo
CodTienda	Código de la Tienda	Entero
Contraseña Cambio Precio	Contraseña actual para realizar cambio de precio	Texto

Habilitado	Si está habilitado que si cambian de precio les pida contraseña	Booleano

Coordinador

Contiene todos los coordinadores dados de alta en el sistema. Sirve para comprobar que el usuario y la contraseña introducidos son correctos en cualquier función del coordinador.

Atributo	Descripción	Tipo
Código Coordinador	Alias del Coordinador	Texto
Nombre Coordinador	Nombre del Coordinador	Texto
Contraseña del Coordinador	Contraseña del Coordinador	Booleano

Relación de Procesos

- **1.1 Insertar Producto**: El sistema inserta en una línea de venta, el producto que haya indicado el vendedor.
- **1.2 Insertar Vendedor**: El sistema comprueba que el vendedor insertado está dado de alta, para poder insertarlo en la cabecera de la venta
- **1.3 Insertar FormaPago**: El sistema genera un nuevo registro en la tabla Multiforma de pago con lo que haya indicado el vendedor
- **1.4 Registrar Venta:** El sistema registra la venta y genera los movimientos de producto asociados.
- **1.5 Crear Reserva**: El sistema genera una nueva cabecera de reserva, para que el vendedor rellene los datos e inserte los productos que quiere reservar el cliente
- **1.6 Crear Cliente**: Una vez que el vendedor haya rellenado los datos correspondientes al cliente, el sistema le da de alta.
- **1.7 Registrar Reserva**: Una vez rellenados todos los datos correspondientes a la reserva, se deben registrar para que en el sistema quede constancia que está pendiente de procesar y preparar los productos reservados.

- **1.8 Terminar Venta**: Una vez que se hayan insertado los productos para facturar, se debe terminar la venta. Entonces, el sistema genera un nuevo registro de la tabla Ticket_TPV, para posteriormente registrar la venta en el proceso 1.4
- **1.9 Terminar Abono**: Una vez que se hayan indicado los productos para abonar, se debe terminar el abono. Entonces, el sistema genera un nuevo registro de la tabla Ticket_TPV, para posteriormente registrar el abono en el proceso 1.13
- **1.10 Cancelar Reserva:** Cuando un cliente ya no está interesado en la reserva que hizo en su momento, se debe cancelar la reserva. En este caso, el sistema da de baja el pedido que se generó en su momento y en compensación con la señal que dejó el cliente, se emite un BONO con el importe correspondiente a la señal, para que pueda ser consumido en compras futuras.
- **1.11 Sincronizar Producto:** Si se ha realizado un cambio urgente en el producto, ya sea cambio de tarifa, de subfamilia o de nombre, se tiene la posibilidad de tener actualizado el producto al momento, haciendo la llamada a esta función.
- **1.12 Convertir Reserva**: En el momento en que los productos están disponibles para la venta, la reserva que se hizo al cliente se puede terminar de facturar. Para ello existe esta opción, que convierte la reserva en un ticket de venta normal, para su posterior registro con el proceso 1.4.
- **1.13 Registrar Abono:** El sistema registra el abono y genera los movimientos de producto asociados.
- **1.14 Consulta Stock:** El sistema permite consultar el stock de un producto en cada uno de los almacenes existentes en el sistema. Cuando se invoca a la función, el sistema devuelve una matriz donde indica el stock disponible para cada almacén.
- **2.1 Apertura Día**: Cuando el encargado pulsa el formulario iniciar día, el sistema genera un nuevo registro en la tabla Caja_Tpv, que corresponde a la caja del día.
- **2.2 Registrar Turno**: Cuando el encargado ha rellenado todos los datos que conciernen al cierre de turno, es decir, recuento de billetes, monedas y ventas en tarjeta, debe registrar el turno. Entonces, el sistema modifica el registro de caja_tpv asignado, para que quede como registrado y cerrado.
- **2.3 Registrar Cierre**: Cuando el encargado ha rellenado todos los datos que conciernen al cierre del día, es decir, recuento de billetes, monedas y ventas en tarjeta debe registrar el turno. Entonces, el sistema modifica el registro de caja_tpv asignado, para que quede como registrado y cerrado. Además, genera movimiento de traspaso de

cuentas, para indicar el dinero que se va a ingresar en el banco y el saldo inicial que quedará para el siguiente día.

- **2.4 Hoja Cierre**: El sistema genera una hoja de Excel con todos los datos correspondientes a la caja, con fecha pasada por parámetro

- **2.5 Registrar_Gasto**: El sistema registra un gasto que ha rellenado debidamente el encargado. Entonces, el sistema genera un movimiento de traspaso a la cuenta de gastos asignada, para saber cuál es el origen del movimiento.

- **3.1 Ventas por Vendedor**: El sistema muestra el importe de las ventas realizadas por el vendedor , indicado por parámetro, dentro del rango de fechas, indicado también por parámetro. También lo muestra en Excel

- **3.2 Ventas por familia**: El sistema muestra el importe y la cantidad vendidas de las subfamilias, indicadas por parámetro, dentro del rango de fechas, también indicado por parámetro. También lo muestra en Excel

- **3.3 Ventas por Producto**: El sistema muestra la cantidad vendida por un producto, indicado por parámetro, dentro de un rango de fechas, también pasado por parámetro. También lo muestra en Excel

- **3.4 Mejores Ventas**: El sistema muestra los mejores productos vendidos según el criterio indicado: Familia, TipoCriterio, NúmeroCodigosaMostrar. También lo muestra en Excel.

- **4.1 Insertar Producto**: El sistema inserta en una línea de transferencia el producto indicado por el encargado.

- **4.2 Registrar Pedido:** El sistema registra el pedido y manda orden a la CENTRAL de que tiene un pedido pendiente de gestionar.

- **4.3 Cambiar Contraseña**: El sistema cambia la contraseña de cambios de precio, por la que haya indicado el encargado que será la nueva.

- **4.4 Registrar Robo:** El sistema registra la transferencia de robos y genera los movimientos de productos asociados. En este caso, es dar de baja del almacén la cantidad de producto indicada por el encargado

- **4.5 Registrar Entrega:** El sistema registra la transferencia de coordinador y genera los movimientos de productos asociados. En este caso, es dar de baja del almacén la cantidad de producto indicada por él y darlo de alta en el almacén asociado al coordinador.

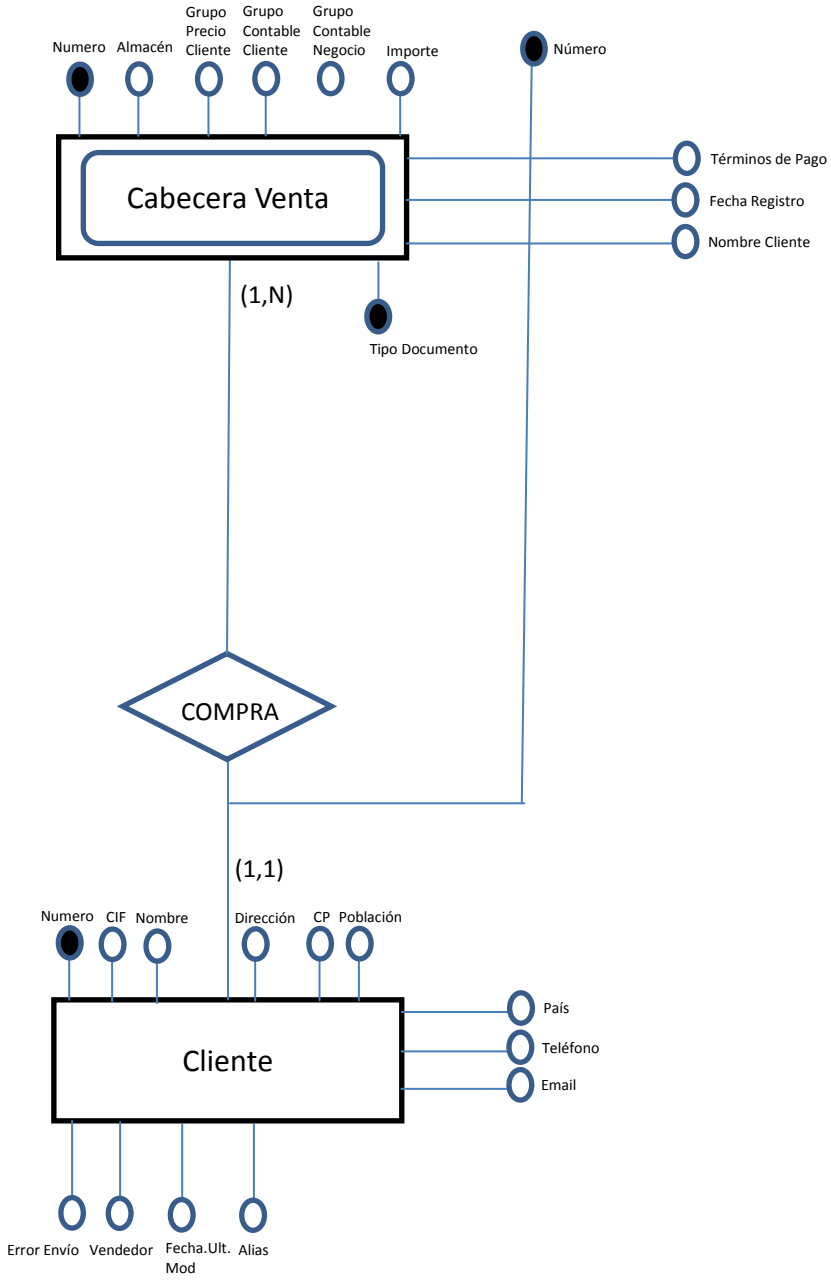
- **4.6 Validar Usuario**: El sistema comprueba que el usuario y contraseña introducidos por el coordinador son correctos. El sistema accede a la tabla coordinadores para comprobar esos datos.
- **4.7 Registrar Recogida**: El sistema registra la transferencia de coordinador y genera los movimientos de productos asociados. En este caso, es dar de baja del almacén asociado al coordinador la cantidad de producto indicada por él y darlo de alta en el almacén asociado a la tienda destino
- **4.8 Validar Autorización**: El sistema comprueba que los datos introducidos por el encargado para validar la transferencia son correctos. El sistema accede a la tabla autorización y comprueba esos datos
- **4.9 Registrar Transferencia:** El sistema registra la transferencia y genera los movimientos de productos asociados. En este caso, es dar de baja del almacén la cantidad de producto indicada por el encargado y darlo de alta en el almacén devoluciones.

5.2.2 Base de Datos

En este apartado, se habla de cómo está configurada la base de datos del programa y de que tablas consta. Por otro lado, se explica, tanto de manera gráfica como de manera escrita, las relaciones existentes entre cada una de las tablas qué componen la base de datos.

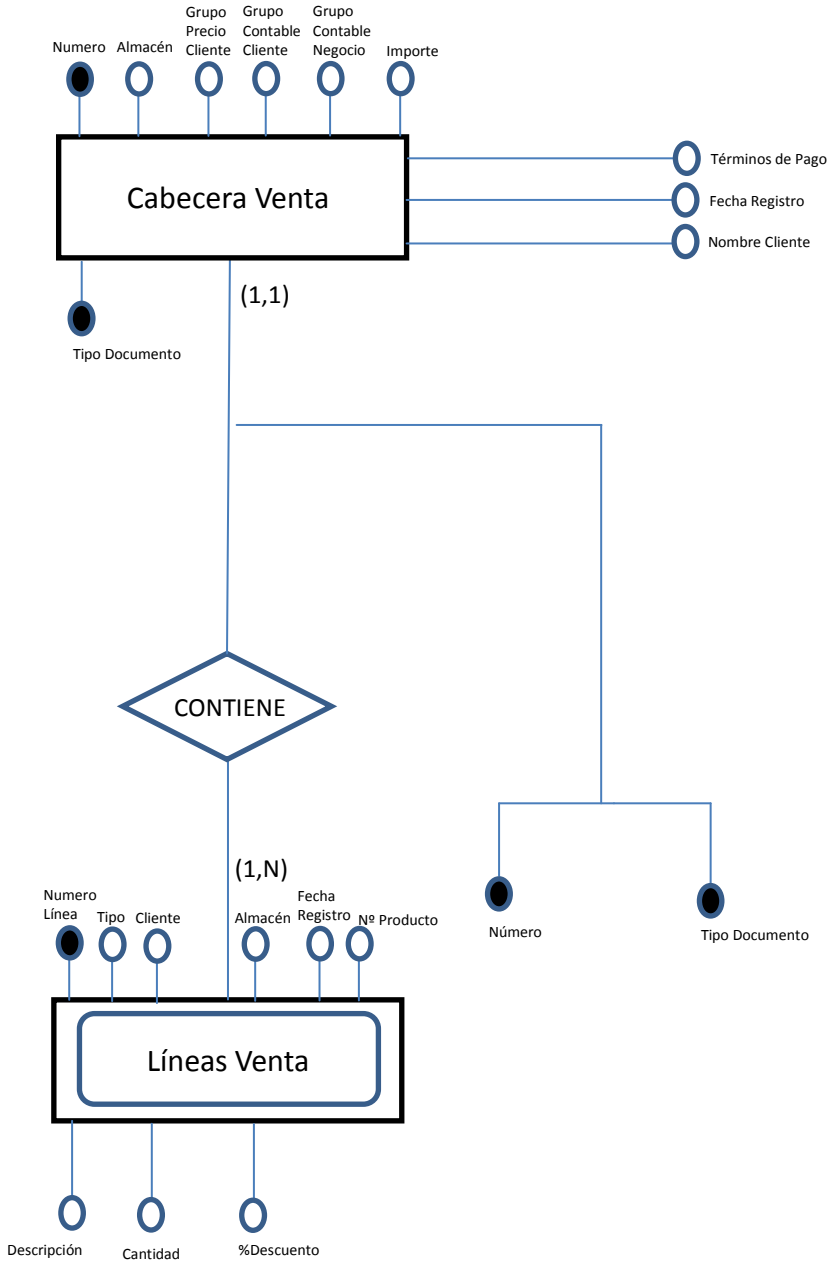
En la tabla Cabecera Venta existe una jerarquía, perteneciente al tipo de documento sobre el que se está trabajando. En este caso, como se hacen una serie de chequeos a la hora de insertar, modificar o borrar registros de la tabla, no es necesario realizar subtipos para cada tipo de documento, a pesar de estar en interrelaciones distintas. Además, el tipo de documento está dentro de la clave primaria para que el control de ese campo sea mayor y no se produzcan incongruencias.

En la tabla Cabecera Reposición existe una jerarquía, perteneciente al tipo de documento sobre el que se está trabajando. En este caso, como se hacen una serie de chequeos a la hora de insertar, modificar o borrar registros de la tabla, no es necesario realizar subtipos para cada tipo de documento, a pesar de estar en interrelaciones distintas. Además, el tipo de documento está dentro de la clave primaria para que el control de ese campo sea mayor y no se produzcan incongruencias.



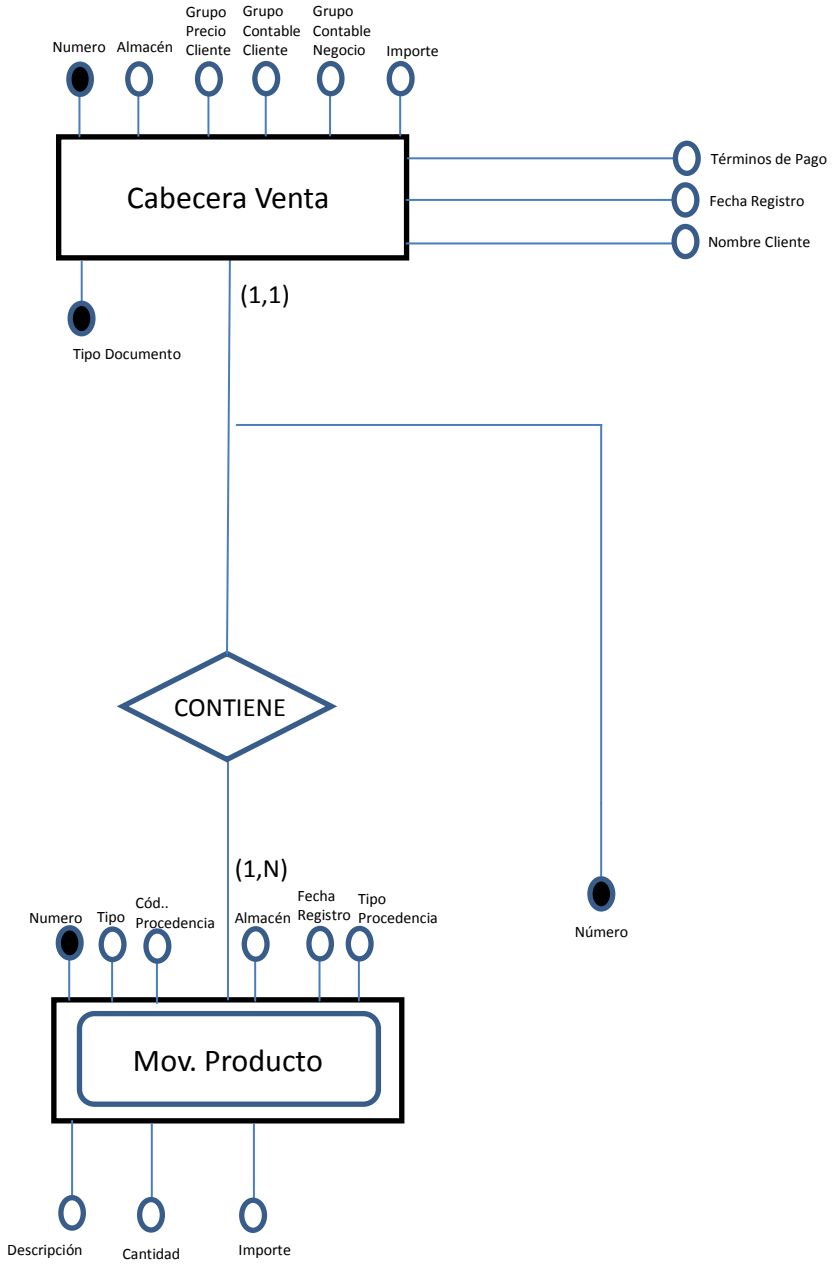
En este caso, la interrelación entre Cliente y Cabecera venta es 1:N, debido a que el cliente puede realizar entre 1-N Compras y en la cabecera venta sólo puede existir un cliente.

Por otro lado, de la interrelación no sale ningún atributo. Por lo tanto, al ser cabecera venta una entidad débil con respecto a cliente, dentro de la clave de cabecera venta se incluirá la clave primaria de Cliente, para saber en todo momento que ventas se le han realizado.



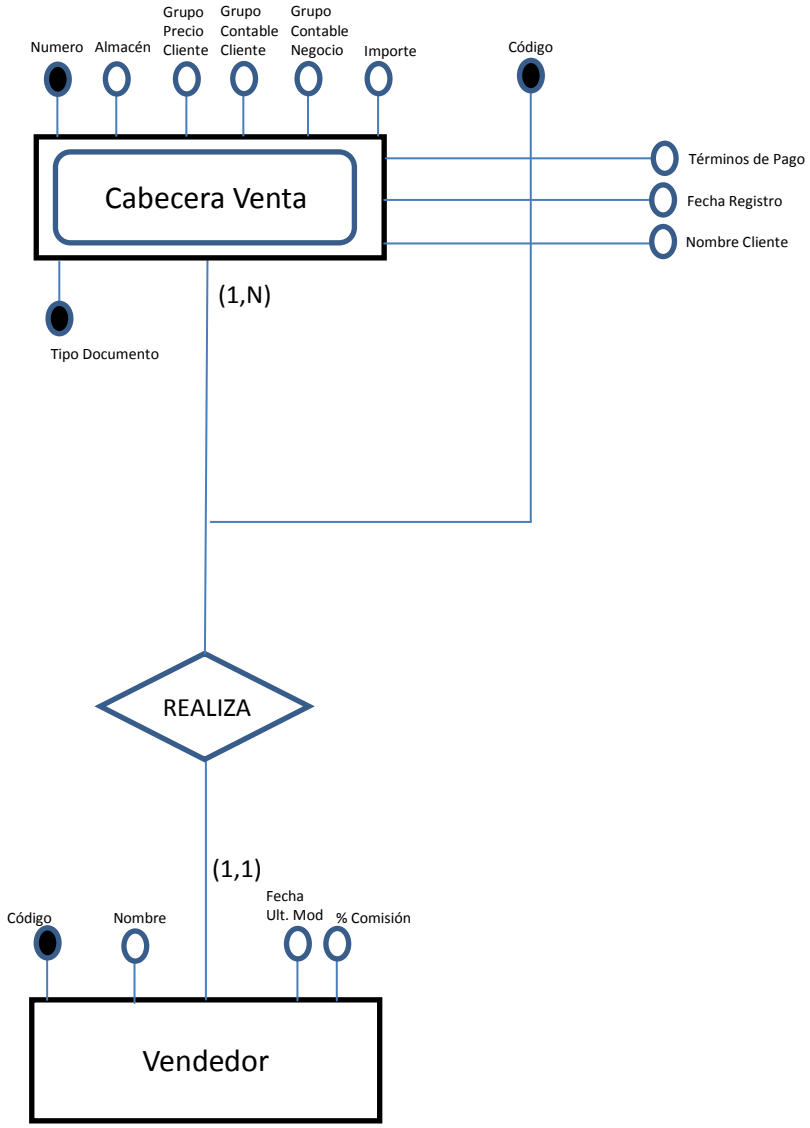
En este caso, la interrelación entre Líneas Venta y Cabecera venta es 1:N, debido a que la cabecera puede tener entre 1-N líneas y la línea venta sólo puede estar asociada a una cabecera venta.

Por otro lado, de la interrelación no sale ningún atributo. Por lo tanto, al ser líneas venta una entidad débil con respecto a cabecera, dentro de la clave de líneas venta se incluirá la clave primaria de cabecera venta, para saber en todo momento que líneas están incluidas en esa cabecera venta.



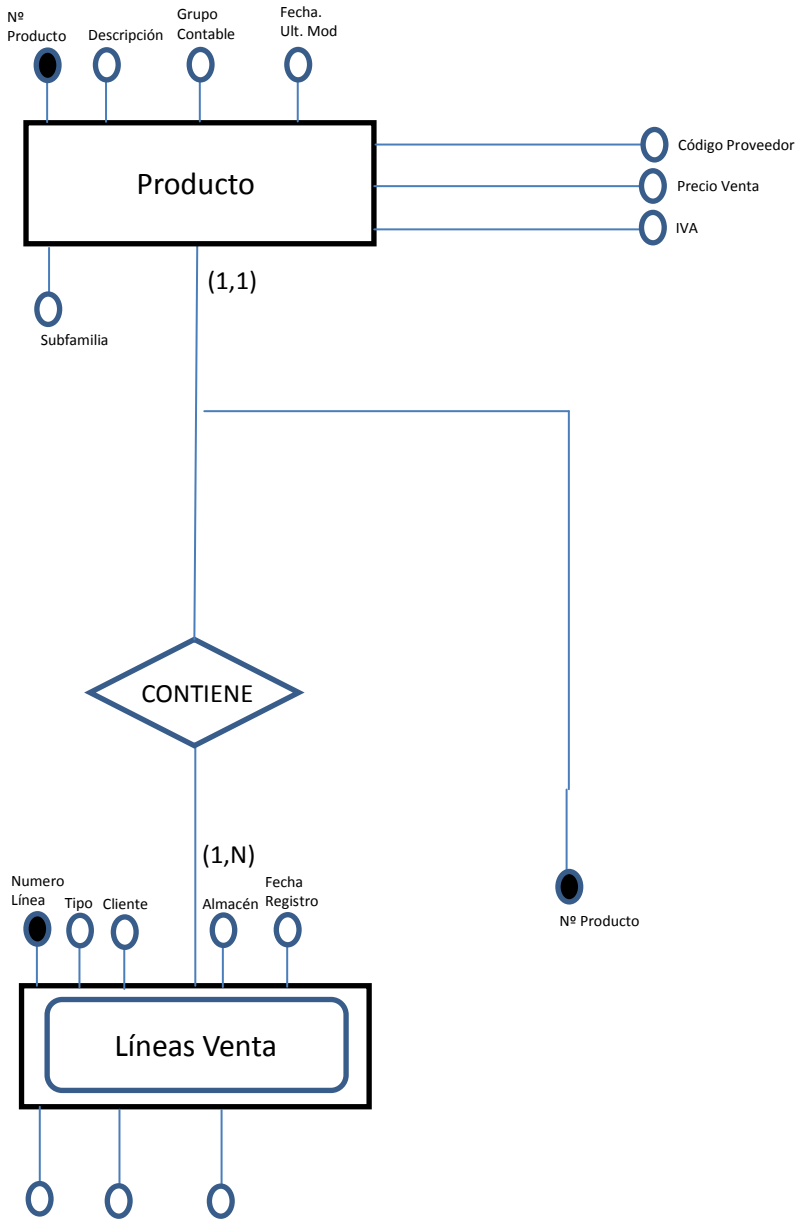
En este caso, la interrelación entre Mov. Producto y Cabecera venta es 1:N, debido a que la cabecera puede tener entre 1-N movimientos de producto y el movimiento de producto sólo puede estar asociado a una cabecera venta.

Por otro lado, de la interrelación no sale ningún atributo. Por lo tanto, al ser Mov. Producto una entidad débil con respecto a cabecera, dentro de la clave de Mov. Producto se incluirá la clave primaria de cabecera venta, para saber en todo momento que movimientos están relacionados con esa cabecera venta.



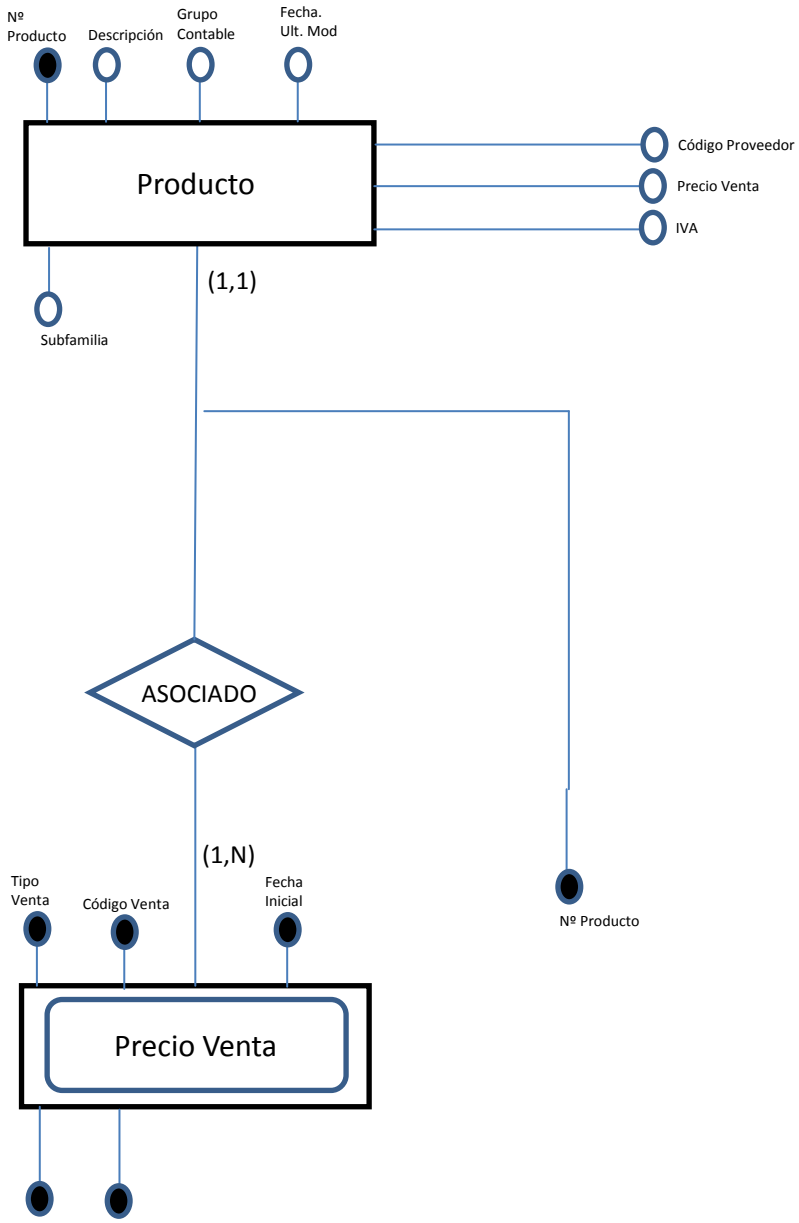
En este caso, la interrelación entre Vendedor y Cabecera venta es 1:N, debido a que el vendedor puede realizar entre 1-N ventas y en la cabecera venta sólo puede existir un vendedor.

Por otro lado, de la interrelación no sale ningún atributo. Por lo tanto, al ser cabecera venta una entidad débil con respecto a vendedor, dentro de la clave de cabecera venta se incluirá la clave primaria de vendedor, para saber en todo momento que ventas ha realizado el vendedor.



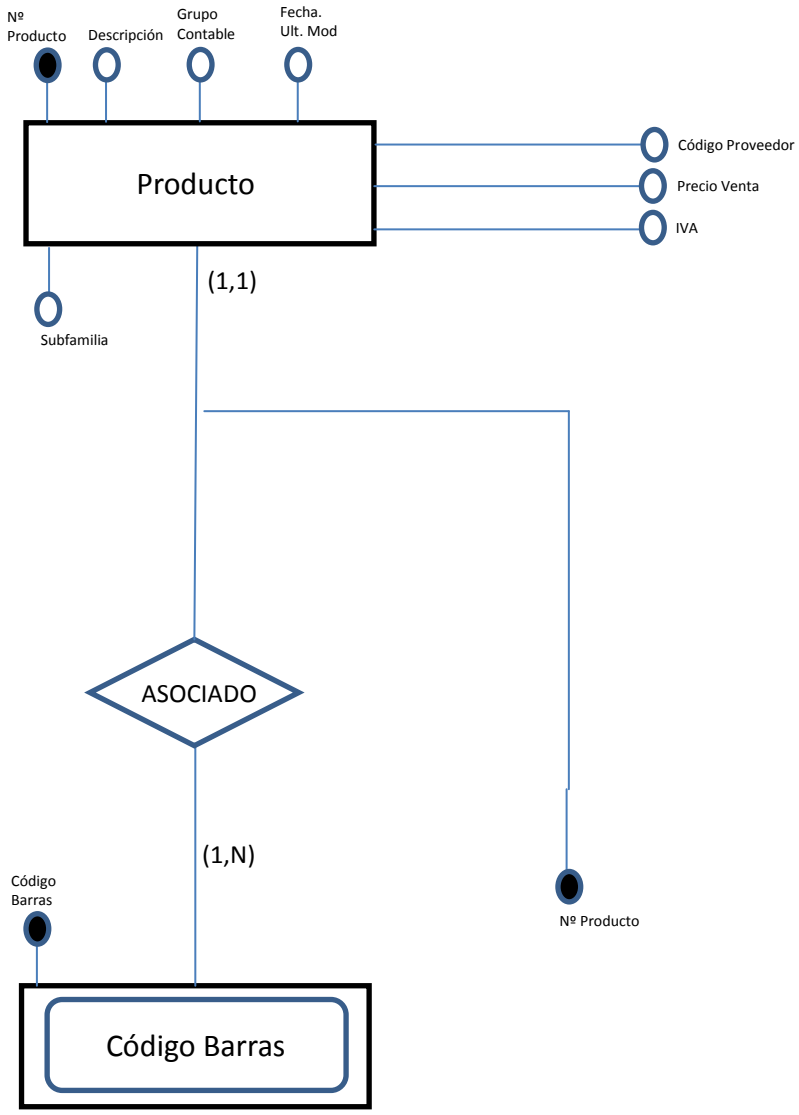
En este caso, la interrelación entre Líneas Venta y Productos es 1:N, debido a que los productos pueden estar en 1-N líneas y en la línea venta sólo puede estar asociada a un producto.

Por otro lado, de la interrelación no sale ningún atributo. Por lo tanto, al ser líneas venta una entidad débil con respecto a producto, dentro de la clave de líneas venta se incluirá la clave primaria de producto, para saber en todo momento en que líneas de venta está inmerso un producto determinado



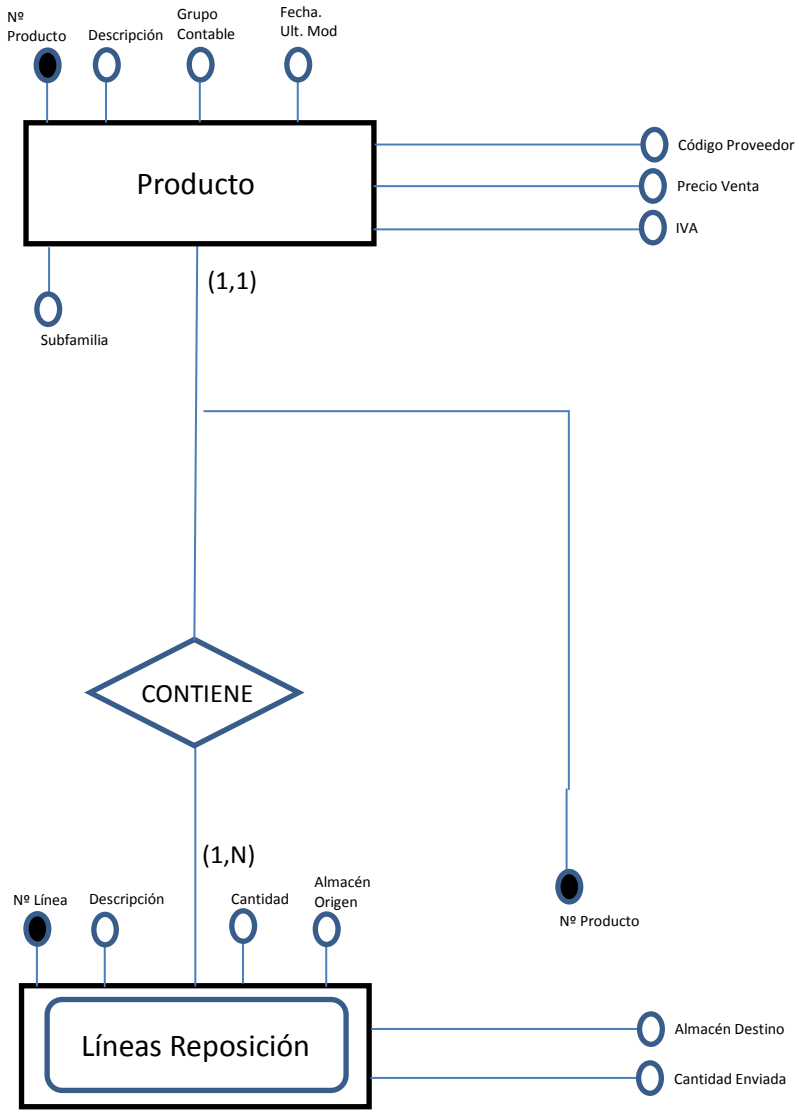
En este caso, la interrelación entre Precio Venta y Productos es 1:N, debido a que los productos pueden tener asociados entre 1-N precios de venta y ese precio de venta sólo puede estar asociado a un producto.

Por otro lado, de la interrelación no sale ningún atributo. Por lo tanto, al ser precio venta una entidad débil con respecto a producto, dentro de la clave de precio venta se incluirá la clave primaria de producto, para saber en todo momento que tarifas de venta tiene asociadas el producto.



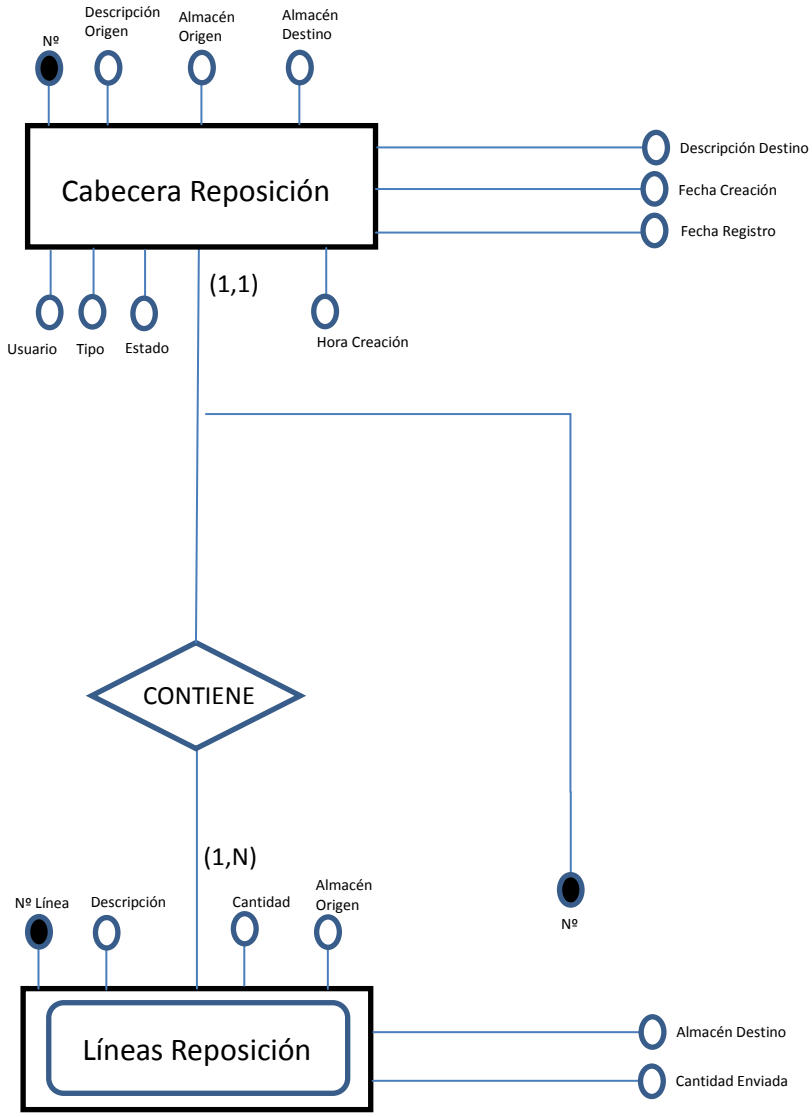
En este caso, la interrelación entre Código barras y Productos es 1:N, debido a que los productos pueden tener asociados entre 1-N códigos de barras y el código de barras sólo puede estar asociada a un producto.

Por otro lado, de la interrelación no sale ningún atributo. Por lo tanto, al ser precio venta una entidad débil con respecto a producto, dentro de la clave de código de barras se incluirá la clave primaria de producto, para saber en todo momento que códigos de barras tiene asociado el producto.



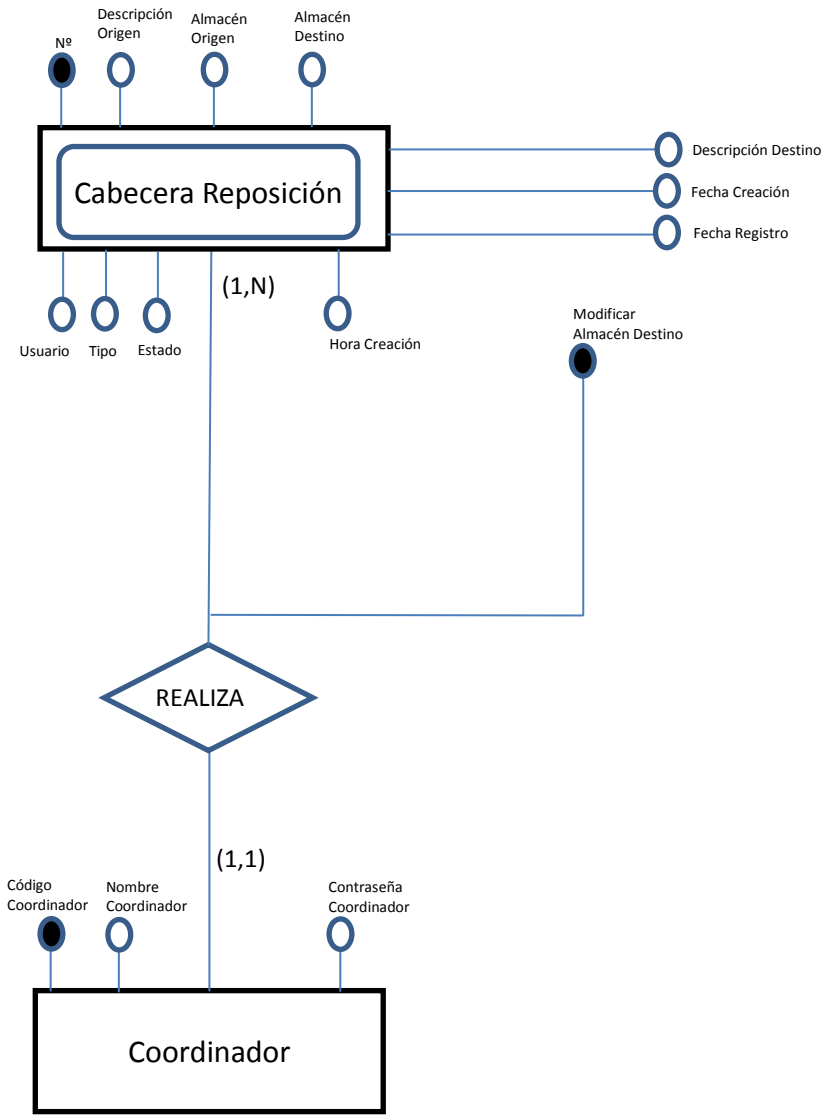
En este caso, la interrelación entre Líneas Reposición y Productos es 1:N, debido a que los productos pueden estar en 1-N líneas de reposición y la línea de reposición sólo puede estar asociada a un producto.

Por otro lado, de la interrelación no sale ningún atributo. Por lo tanto, al ser líneas reposición una entidad débil con respecto a producto, dentro de la clave de líneas reposición se incluirá la clave primaria de producto, para saber en todo momento en que líneas de reposición está el producto.



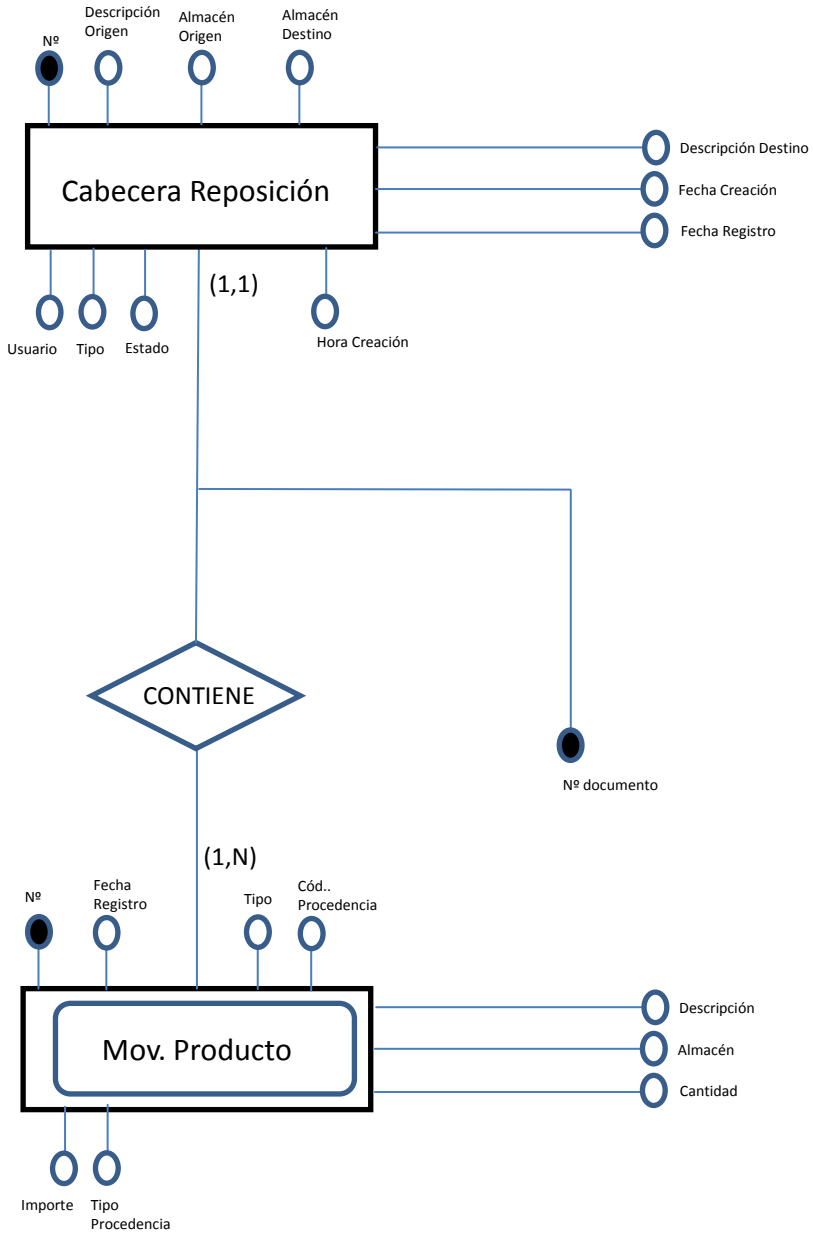
En este caso, la interrelación entre Cabecera de Reposición y Líneas Reposición es 1:N, debido a que la cabecera de reposición puede tener entre 1-N líneas de reposición y además, en esa línea de reposición sólo puede estar asociada a una cabecera de reposición.

Por otro lado, de la interrelación no sale ningún atributo por lo tanto, al ser líneas reposición una entidad débil con respecto a cabecera reposición, dentro de la clave de líneas reposición se incluirá la clave primaria de cabecera de reposición, para saber en todo momento que líneas de reposición componen una cabecera de reposición.



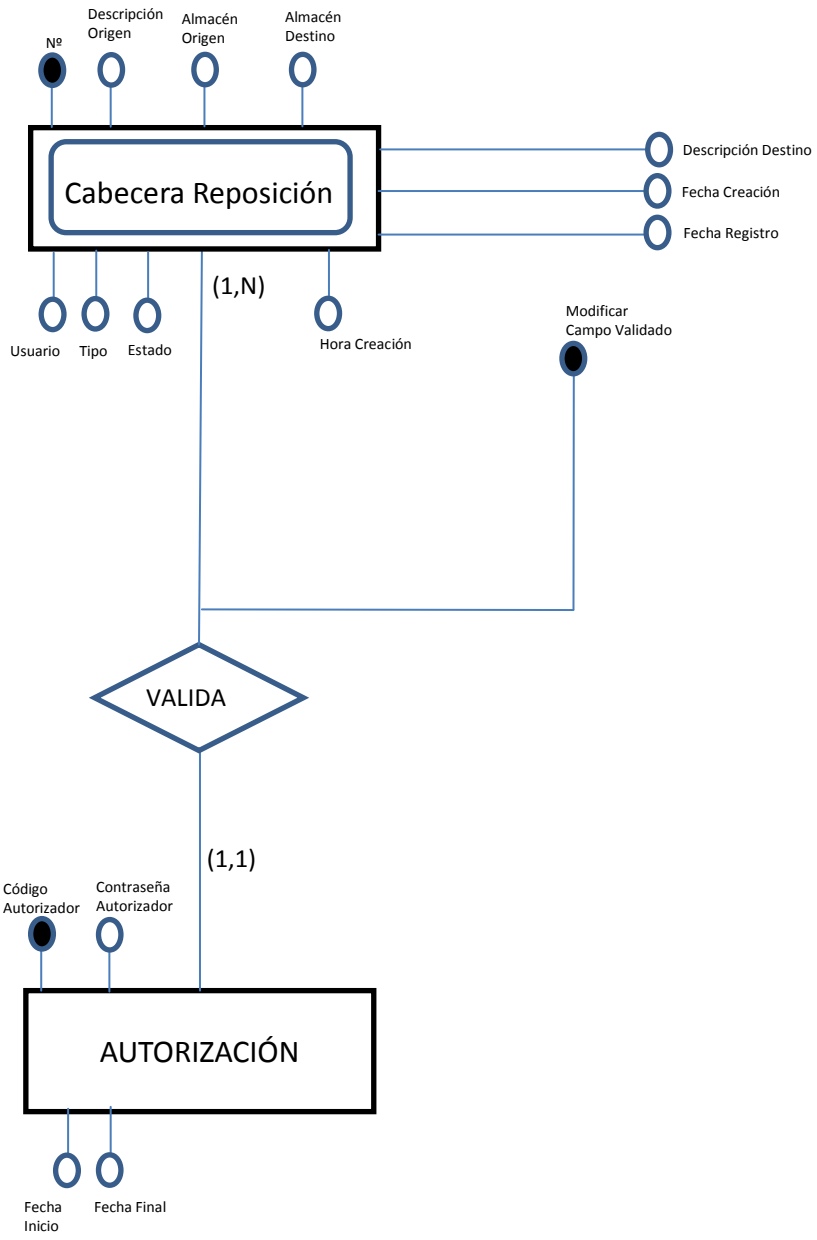
En este caso, la interrelación entre Coordinador y Cabecera de Reposición es 1:N, debido a que el coordinador puede realizar entre 1-N cabecera reposición y además, en esa cabecera reposición sólo puede estar asociada a un coordinador.

Por otro lado, de la interrelación no sale ningún atributo por lo tanto, al ser cabecera reposición una entidad débil con respecto a coordinador, en este caso lo que se hace es modificar el almacén destino para que la mercancía que se registre se cargue en el stock del coordinador.



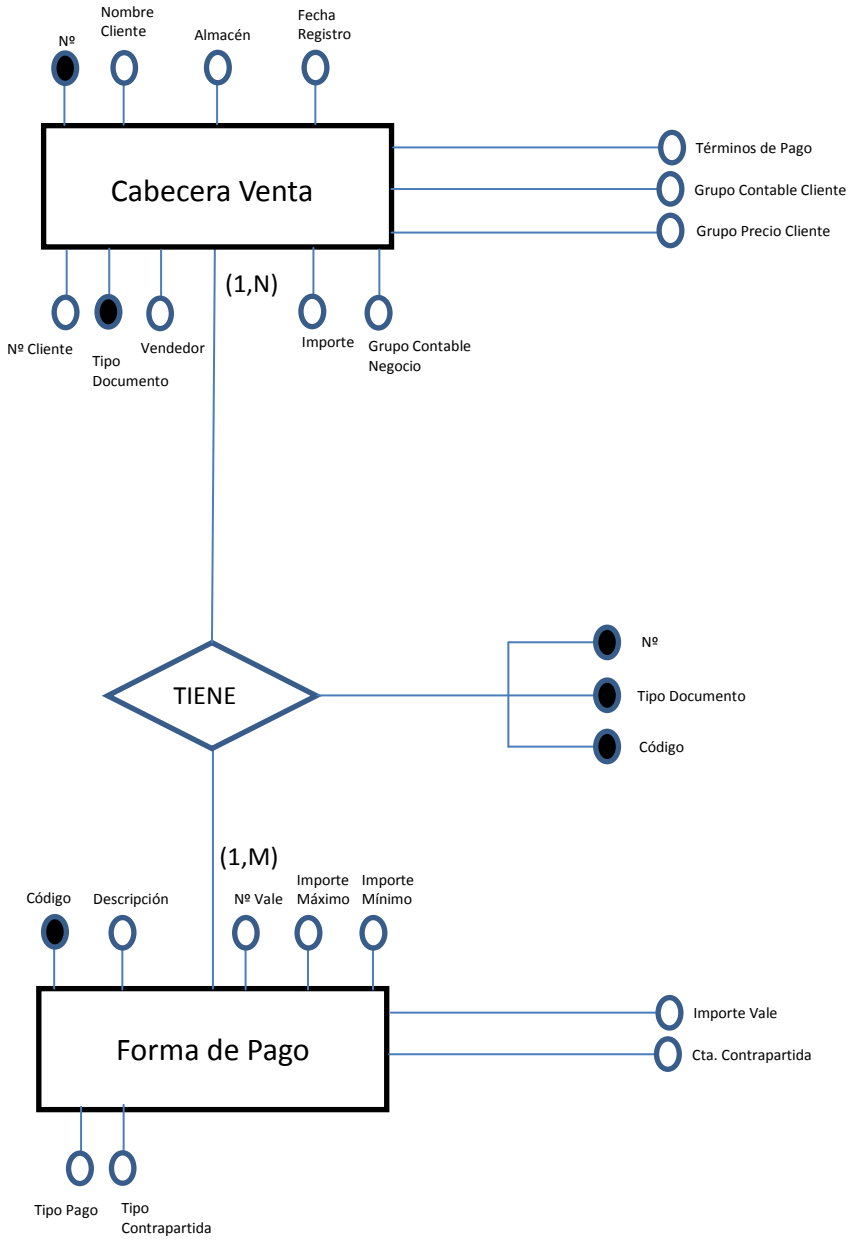
En este caso, la interrelación entre Cabecera de Reposición y Mov. Producto es 1:N, debido a que la cabecera de reposición puede estar en 1-N Mov. Producto y el Mov. Producto sólo puede estar asociado a una cabecera de reposición.

Por otro lado, de la interrelación no sale ningún atributo. Por lo tanto, al ser mov. Producto una entidad débil con respecto a cabecera reposición, dentro de la clave de mov. producto se incluirá la clave primaria de cabecera de reposición, para saber que movimientos de productos se han realizado a través de esa cabecera de reposición.

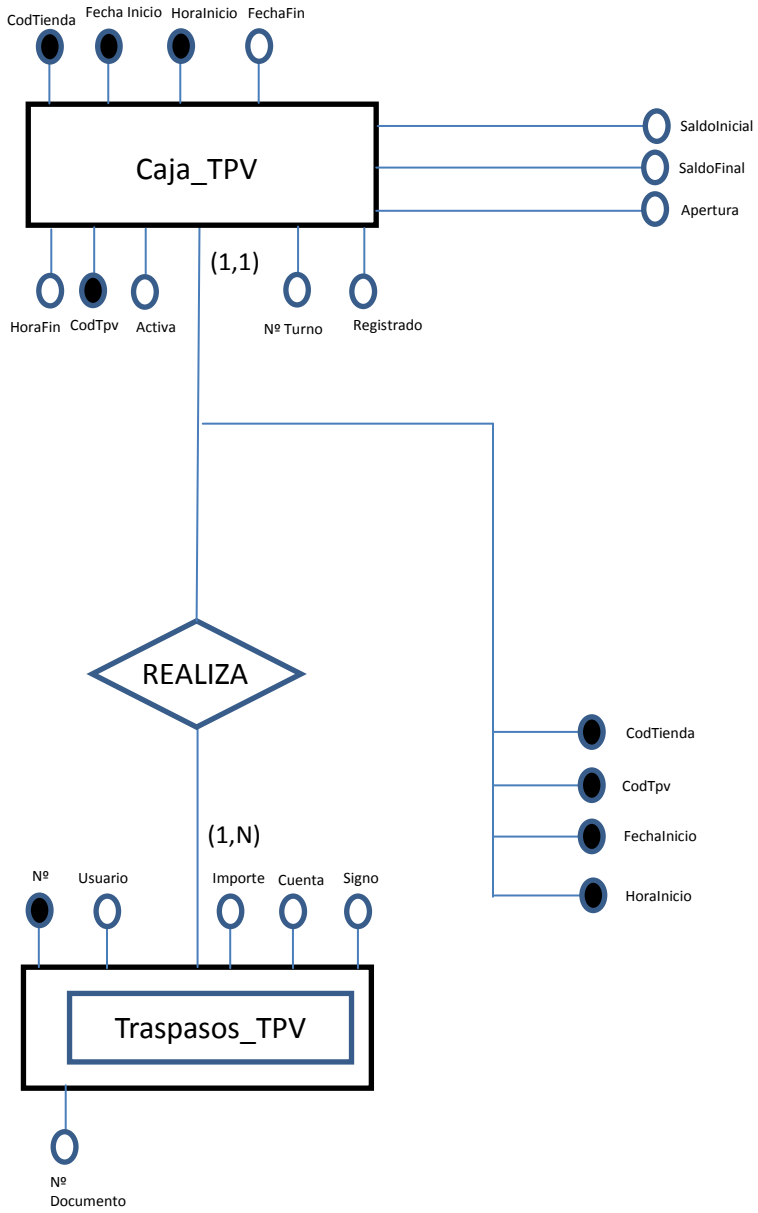


En este caso, la interrelación entre Autorización y Cabecera de Reposición es 1:N, debido a que una autorización puede validar entre 1-N cabecera reposición y la cabecera reposición sólo puede validarse una sola vez.

Por otro lado, de la interrelación no sale ningún atributo. Por lo tanto, al ser cabecera reposición una entidad débil con respecto autorización, en este caso lo que se hace es modificar el campo autorizado para que a la hora de registrar esa transferencia, sepa que esta previamente autorizada.



En este caso, al ser la relación N:M se crea una nueva tabla que será Multiforma de Pago, que en la clave primaria se guardará la clave primaria de cabecera venta (Nº,TipoDocumento) y la clave primaria de Forma de pago(Código). Además, la tabla multiforma de pago tendrá otros atributos adicionales.



En este caso, la interrelación entre Caja_tpv y Traspasos_TPV es 1:N, debido a que durante el día se pueden realizar varios traspasos de dinero de una cuenta a otra y el documento sólo puede estar relacionado con la caja del día en que se registró.

Por otro lado, de la interrelación no sale ningún atributo. Por lo tanto, al ser traspasos_tpv una entidad débil con respecto a caja_tpv, dentro de la clave traspasos_tpv se incluirá la clave primaria de caja_tpv, para saber en todo momento que traspasos de dinero se han realizado durante todo el día.

A continuación, se explican los chequeos, inserciones y disparadores que existen en cada una de las tablas, para saber que controles se hacen sobre los campos para que insertar, modificar o borrar no produzca desajustes en la base de datos.

CHECKS

- En la tabla Ticket_TPV tiene varios checks, que son: en tipodocumento se ha puesto que las opciones sean factura, devolución o reserva. En tipo pago se ha puesto que las opciones sean tarjeta, vale o caja. En origen devolución se han puestos que las opciones sean Caja o Tarjeta.
- En la tabla Movimiento Producto tiene varios checks: en tipomovimiento se ha puesto que las opciones sean venta, transferencia. En tipoprocedencia las opciones se han puesto que sean Cliente, Producto
- En la tabla Cabecera Reposición tiene un check en el campo Tipo, se ha puesto que las opciones sean Pedido, TransferenciaCoordinador, Transferencia de Robos, Transferencia.
- En la tabla Líneas Venta tiene varios checks, que son: en tipodocumento se ha puesto que las opciones sean factura, pedido, abono. En tipo se ha puesto que las opciones sean Producto o vacío, por si se quiere añadir algún comentario a la factura.
- En la tabla Cabecera Venta tiene un check en el campo tipodocumento para que las opciones a introducir sean Factura, Abono o Pedido.
- En la tabla Precio Venta tiene un check en el campo tipoventa para que las opciones a elegir sean Grupo Precio Cliente o Cliente.
- En la tabla Multiforma de Pago tiene los siguientes checks: en el campo tipodocumento las opciones a introducir son: factura, devolución o Reserva. Por otro lado, en tipoContrapartida las opciones disponibles son Cuenta o Banco.

INSERCIONES

- En la tabla cliente cuando se quiere insertar un nuevo registro, el sistema realiza una serie de comprobaciones previas, que son las siguientes: En primer lugar, busca el número de cliente que se le debe asociar, a partir del número de serie asignado a los clientes. A continuación, comprueba que en la base de datos se ha configurado una ficha cliente, en caso negativo el sistema emite un mensaje de error. Por otro lado, si se quiere

borrar un cliente, el sistema comprueba si tiene movimientos asociados, en caso de tenerlos impide borrarlo.

- En la tabla Caja_TPV cuando se quiere insertar un nuevo registro, el sistema realiza una serie de comprobaciones previas, que son las siguientes: Comprueba que la caja del anterior día está cerrada, en caso negativo el sistema emite un mensaje de error indicando ese hecho. Por otro lado, comprueba que el número de turno de la caja no sea mayor que el número máximo de turnos. Además, si existe alguna caja activa asociada a esa caja, el sistema emite un mensaje de error indicando que se ha realizado la apertura del día.
- En la tabla Cabecera Reposición cuando se quiere insertar un nuevo registro, el sistema comprueba qué número debe asignarle al nuevo registro, a partir del número de serie asignado.
- En la tabla Linea Venta cuando se inserta un nuevo registro, el sistema comprueba si el producto indicado en ese registro está asociado alguna promoción vigente. En caso afirmativo, se rellenan una serie de datos para que luego se pueda aplicar la promoción.
- En la tabla Cabecera Venta cuando se quiere realizar la inserción de un nuevo registro, el sistema realiza las siguientes comprobaciones: Si se va eliminar un registro, se comprueba si el ticket está asociado a una reserva. Si se cumple ese requisito, el sistema no deja borrar el registro, ya que existe un importe pagado cómo señal.
- Si en la tabla Multiforma de Pago se quiere eliminar un registro, el sistema debe comprobar si esa forma de pago está registrada. En caso afirmativo, emite un mensaje de error indicando que no se puede borrar el registro. Por otro lado, si el registro que se quiere eliminar es la primera forma de pago insertada en la venta, el sistema impide borrar ese registro. Además, si el registro que se quiere borrar tiene forma de pago tarjeta y está autorizada, el sistema impide eliminarlo.
- En la tabla traspasos_tpv cualquier registro que se quiera borrar, el sistema, directamente, emite un mensaje de error impidiendo su borrado.

DISPARADORES

- En la tabla Ticket_TPV se debe verificar que la forma de pago introducida existe dentro de la base de datos. Por otro lado, dependiendo de la forma de pago introducida, se rellenan una serie de campos.

- Si en la tabla Caja_tpv se quiere realizar una modificación en algún campo, se debe verificar si la caja no está registrada.
- En la tabla Cabecera Reposición se debe verificar si el destino o el origen introducido está dado de alta en la base de datos. Además, se debe verificar que tanto el almacén destino cómo el de origen no estén cerrados. Por otro lado, una vez verificado que todos los campos están correctos, el sistema modifica en todas las líneas de reposición, asociadas a la cabecera, el campo destino. Por último, dependiendo del tipo de reposición que se quiera insertar, el sistema realiza la llamada a una función.
- En la tabla Linea Venta se debe verificar que el producto insertado está dado de alta en la base de datos. Además, si la línea de venta es de tipo abono y asociada a una venta, verifica si la cantidad ha sido abonada previamente. Por otro lado, se debe verificar que el precio modificado, de forma manual, no sea negativo.
- En la tabla Cabecera Venta se verifica en los registros de tipo reserva que el cliente asociado no es un cliente genérico. Por otro lado, si el registro es de tipo abono y se va hacer contra factura, se debe comprobar que no hay líneas de venta creadas con anterioridad.
- En la tabla Códigos de Barras se debe verificar que el código de barras insertado no esté asociado a otro producto.
- En la tabla Multiforma de Pago se debe realizar una serie de verificaciones: que el código de la forma de pago no sea vacío, que no se pueden combinar distintas formas de pago de tipo vale, que no existan dos registros con la forma de pago CAJA, que la forma de pago introducida exista en la base de datos, que si la forma de pago tiene importe mínimo e importe máximo, se verifique que el importe introducido esté dentro del rango, que el número de vale introducido tenga el formato correcto y no esté en otro registro dentro de la misma venta.
- En la tabla Linea Reposición se debe verificar que el producto insertado esté dado de alta en la base de datos.

Por otro lado, cabe destacar que el motor utilizado es SQL Server, debido a que la optimización de los recursos es muy buena y eso a la hora de trabajar con muchos usuarios, de forma simultánea, es un punto a su favor. A continuación, se explican las principales características de SQL Server:

- El motor de la base de datos es compatible con varias plataformas de Windows, que pueden ser desde Windows 98 hasta Windows 7, en caso de usuarios y de Server 2000 hasta server 2008, en caso de los servidores.
- El motor de la base de datos protege la integridad de los datos a la vez que minimiza la carga de datos, para poder administrar de manera eficiente la modificación de la base de datos por varios usuarios de manera simultánea. Además, las consultas permiten hacer referencias a datos externos, como si fuesen parte de la base de datos.
- SQL Server incluye un conjunto de herramientas administrativas y desarrollo que mejora el proceso de instalación y uso del SQL. Por otro lado, tiene un modelo de programación que se integra de forma sencilla con WINDOWS, haciendo posible que el uso de la base de datos se haga de forma fluida. Esto permite que los clientes puedan implementar un SQL con un trabajo mínimo
- SQL permite extraer y analizar datos, para realizar un procesamiento en línea sin que afecte lo más mínimo a la base de datos.
- Las aplicaciones de SQL Server se pueden ejecutar en el propio servidor, ya que la aplicación se conecta a SQL utilizando los componentes de comunicación de WINDOWS, en vez de utilizar la red.
-SQL Server tiene una completa interfaz gráfica, para que la gestión y administración del mismo no sea muy compleja.
- SQL Server es compatible con multitud de herramientas de desarrollo, evitando así que sea un software limitado.
- SQL Server permite realizar copias en línea, para mantener la consistencia de la base de datos.
- Las copias de seguridad permiten realizar recuperación de tablas individuales, evitando tener que restaurar la base entera.
- En SQL Server existe un único ID de login tanto para la red como para la base de datos, para mejorar la seguridad y facilitar la administración.
- SQL Server permite password y cifrado de datos en red, para mejorar la seguridad.

6 Conclusiones

Antes de empezar hablar sobre las principales conclusiones acerca del PFC, voy a comentar la siguiente aclaración:

El título original era: SISTEMA INTEGRADOS DE GESTIÓN (ERP) EN PYMES, ya que se iba hablar de los sistemas ERP más importantes dentro de las PYMES, pero a los meses de empezar el PFC se me brindó la oportunidade de hacer algo práctico en una empresa, por lo que decidí cambiar el título del PFC y añadir una parte práctica al PFC para que fuese más completo. Además, gracias a la parte práctica se puede ver mejor la funcionalidad de los ERP.

El objetivo principal del proyecto es la implantación de un sistema ERP en una empresa de Juguetes. Para ello se han realizado una serie de pasos:

En primer lugar, se debe realizar un estudio previo de todos los movimientos que realiza diariamente la empresa, ya sea de venta, de compra o de almacén. A partir de esos datos, se puede evaluar la magnitud de la empresa.

Una vez recabados esos datos, se debe evaluar qué tipo de sistema ERP es el más idóneo para la empresa, para que pueda soportar ese número de transacciones. En este caso, el sistema más idóneo es Navision, ya que soporta una gran carga de datos y tiene una extensa funcionalidad.

Navision es un sistema gestor de medianas empresas, que permite hacer un seguimiento exhaustivo de la vida de un producto: cuando entró en el almacén, en qué zonas del mismo se ha ubicado, los movimientos para realizar la preparación de pedidos, las ventas realizadas en la tienda en un día en concreto, si ha sido devuelto el producto a CENTRAL. Estos hechos son de gran utilidad para el departamento comercial, ya que pueden conocer la rotación que tiene ese producto, para saber si merece la pena realizar posteriores compras.

Por otro lado, de cualquier registro que se haga en Navision queda constancia del autor y para futuras consultas resulta muy beneficioso, ya que se puede saber las ventas que ha realizado un vendedor en un rango de fechas, parar calcular su comisión. Además, se puede saber qué cantidad y número de líneas ha preparado en un día un operario de almacén, así poder evaluar su rendimiento. En definitiva, se puede hacer un seguimiento de cada uno de los empleados de la empresa y saber que hacen en cada momento.

Además, Navision tiene una serie de premisas primordiales en cuanto a seguridad: No permite a un mismo usuario tener abiertas dos sesiones del mismo programa, para evitar posibles fraudes o errores intencionados.
Navision no permite largos tiempos de inactividad, es decir, si pasado un periodo de tiempo, el usuario no ha realizado nada, el sistema anula la sesión para que otro usuario pueda utilizarla.

Otro beneficio importante dentro de Navision es el motor de base de datos utilizado, en este caso SQL Server, que permite trabajar simultáneamente a un número determinado de usuarios y puedan realizar, de manera paralela, una serie de transacciones sin que los usuarios se vean afectados, a menos que esas transacciones accedan a las mismas tablas.
En este proyecto, el punto primordial es desarrollar un módulo de ventas para las tiendas, que fuera lo más intuitivo y sencillo, ya que el objetivo es no entorpecer la venta y que sea lo más ágil posible, para que el cliente siga confiando en la empresa.

Para ello, se necesitaban unos requisitos previos donde se indica que datos se quieren ver de la tienda y la funcionalidad que se quiera instaurar. Una vez recabada toda esa información, ya se procede al desarrollo del módulo.
En este caso, el sistema desarrollado evita mucho manejo por parte de los operarios, evitando que los errores humanos tengan un alto porcentaje. Para ello, mediante permisos y roles, se ha ido limitando una serie de funciones que realiza el sistema de forma automática.

Como conclusión, se ha creado un módulo de ventas muy intuitivo y muy mecánico, en el sentido que se ha acoplado lo máximo posible a la funcionalidad que ofrece Navision, para que todo esté automatizado y no tener que retocar mucho el estándar.

Por lo tanto, las funciones que realizan los usuarios son: vender, transferir, cotejar, iniciar día, registrar gastos, realizar el cierre del día y realizar pedidos, es decir, funciones básicas dentro del ámbito de la venta.

Gracias a la realización del proyecto, he podido comprender mejor el funcionamiento de un ERP en cuanto a: conocer a fondo el ciclo de vida de un producto, la generación de asientos contables, la centralización de las ventas con su correspondiente comunicación, conocer a fondo la funcionalidad del almacén.

Por otro lado, la práctica ha resultado más asequible de lo que parecía gracias a los conocimientos adquiridos en la universidad, ya que en la práctica he necesitado conocimientos en: Ingeniería del Software, Base de Datos, Seguridad y Auditoría(que me ha aportado una visión práctica).

Además, yo también he hecho mis aportaciones al PFC, como pueden ser: análisis de los ERP, funcinamiento de un ERP y la aplicación realizada en el PFC.

En conclusión, se puede decir que los objetivos del PFC se han cumplido con notoriedad.

En definitiva, se puede asegurar que Navision es un sistema fiable, intuitivo, con una seguridad estable y, sobre todo, que a la hora de navegar facilita mucho la tarea. Esta última característica, aunque parezca que es liviana, es muy importante a la hora de analizar la empresa y ver los rendimientos, para analizar si es necesario un cambio de rumbo en la forma de trabajo, para mejorar el rendimiento y los beneficios de la empresa.

7 Futuras Líneas de Trabajo

A pesar de que Navision como gestor de medianas empresas es bueno, siempre se puede mejorar, para que algunas funciones se realicen de forma más rápida y ágil, asegurando que el acceso a la información se realice sin problemas.

Una de las funcionalidades donde existe un amplio margen de mejora es la coordinación y utilización del motor de la base de datos. En la actualidad, la utilización de cada una de las funciones del motor de la base de datos es ínfima, por lo que repercute negativamente sobre Navision.

Donde más se nota ese hecho es a la hora de realizar consultas o generando informes. Si la consulta se realiza en SQL Server, los resultados son casi inmediatos.En cambio, si se realiza sobre Navision, el resultado depende de la clave que se haya utilizado, es decir, si la clave ha sido la correcta, el tiempo que tarda en mostrar los resultados no es muy grande pero si la clave es incorrecta, el tiempo empleado en mostrar los resultados es muy grande, ya que la ordenación de los datos ralentiza el proceso.

Además, debería haber más coordinación entre los dos programas, es decir, que en Navision se aproveche más de todo lo que ofrece SQL, para que la base de datos no sufra si la carga de datos es muy alta. SQL ofrece una amplía configuración de las tablas. Gracias a esas configuraciones, las consultas, prácticamente, se harían en lenguaje transact SQL, que ha quedado demostrado que se obtienen resultados instantáneos. Este punto se debería mejorar en un futuro, porque lo principal a la hora de evaluar el rendimiento de una empresa es el acceso a los datos.

Otro punto mejorable es el tema de las claves y su reconstrucción. Existe una función, por la cual se regeneran las claves de una tabla, haciendo que cualquier consulta se pueda realizar en el menor tiempo posible. En la actualidad, al no ser un proceso automático, si no se realiza periódicamente, repercute en la ordenación y acotación de una tabla, al estar desestructurada la clave.

Por otro lado, otro aspecto que se debería mejorar, en cuanto a facilidad y accesibilidad, es el tema de roles y permisos para cada uno de los usuarios de la base de datos. La configuración de los roles se debe hacer tanto en Navision como en SQL Server, ya que los roles se asignan en Navision y los permisos en SQL Server. En SQL Server se debe decidir que usuarios deben ser “propietarios” de la base de datos, para navegar por cualquier rincón. Además, en SQL se debe decidir si al usuario se le asigna permiso de modificación, inserción, eliminación, etc dentro de las tablas.

En cambio, en Navision se deben asignar los roles, para decidir que funciones pueden realizar o a que formularios pueden acceder. Además, dentro de cada formulario decidir que campos pueden editar .

Otro punto importante es mejorar la comunicación de las tiendas. Como se ha explicado anteriormente, se comunican mediante servicio Web con la central. Esta comunicación se realiza mediante un conector, que se instala en el servidor de la tienda.

Para determinadas funcionalidades esa comunicación es satisfactoria, ya que no hay confrontación con otros usuarios y, por lo tanto, es suficiente para enviar los datos de forma regular. En cambio, existen otras funciones que el canal de comunicación es insuficiente, debido a que a la tabla acceden muchos usuarios, haciendo que la consulta o actualización de los datos sea lenta.

Donde más se nota es en la actualización diaria de los productos, ya que si el día anterior se crearon o modificaron un número elevado de productos, el tiempo de ejecución es muy elevado, debido a que las tiendas realizan la apertura del día casi a la

misma hora, por lo que a la tabla de productos están accediendo un número elevado de usuarios.

Como se ha explicado anteriormente, tanto la petición como la respuesta del servicio se realiza mediante ficheros XML. Entonces, una alternativa para mejorar la sincronización de productos, es generar en CENTRAL un fichero de productos para cada día y la tienda, mediante FTP, pueda acceder a ese fichero y copiarlo en el ordenador, para realizar la actualización de forma local y agilizar el proceso.

Otra opción es generar una base de datos espejo, donde sólo accedan las tiendas y allí se inserten todos los movimientos de las tiendas. Dicha base espejo estará en contacto con la base principal. Entonces, cada cierto tiempo, intercambiarán la información que se haya ido generando en cada una de ellas, para que los datos sean idénticos.

En conclusión, a partir de este PFC se pueden realizar otros, que vayan enfocados a mejorar la funcionalidad. En este caso, existen varias alternativas:

- SQL Server: El PFC esté enfocado a mejorar la coordinación entre SQL Server y Navision, facilitando la tarea de configuración de cada uno de ellos.
- Comunicación: El PFC esté enfocado en la mejora de la comunicación entre CENTRAL y las tiendas, facilitando el volcado de los datos.

8 Bibliografía

Páginas en la red (Información ERP)

http://www12.sap.com/spain/solutions/business-suite/features-functions/index.epx

http://www.dynamicsit.com.co/pdf/modulos/Guia%20Rapida%20de%20MS%20Dynamics%20AX.pdf

http://www.microsoft.com/dynamics/es/es/products/nav-overview.aspx

Páginas en la red (Lenguaje de programación)

http://www.slideshare.net/makac0/gua-rpida-lenguaje-cal

http://www.slideshare.net/pabloesp/programacin-con-cal-para-microsoft-business-solutions-navision

Páginas en la red (Descargar Cliente de Navision)

http://dynamicsuser.net/media/p/85325.aspx

Imágenes del Proyecto

Las imágenes mostradas en el proyecto han sido sacadas tanto de la demostración que hace Navision en su página como de la aplicación desarrollada.

Norma

Norma ISO/IEC 17799:2005. Esta Norma ha sido utilizada para poder tratar todo lo referente a la seguridad en Navision, ya que se debe regir por esta norma.

Libros

SQL Server 2005 / Francisco Charte Ojeda (2006): Este libro explica muy bien las características y funcionalidad del motor SQL.

9 Presupuesto

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:

Miguel Ángel Carbonero Jimenez

2.- Departamento:

Informatica

3.- Descripción del Proyecto:

- Titulo: Implantación de un sistema ERP en una juguetería
- Duración (meses): 22

Tasa de costes Indirectos: **20%**

4.- Presupuesto total del Proyecto (valores en Euros):

61.052,00 Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a)]	Coste hombre mes	Coste (Euro)
Carbonero Jimenez, Miguel Angel	49000002C	Ingeniero Junior	22	1.500,00	33.000,00
Lopez Rubio, Ignacio	47685678X	Ingeniero Senior	6	2.500,00	15.000,00
					0,00
					0,00
					0,00
		Hombres mes	**28**	**Total**	48.000,00

[a)] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d)]
Servidor HP ProLiant DL380 G7	6.000,00	60	22	60	1.320,00
Ordenador HP Pro 3120	700,00	100	22	60	256,67
		100		60	0,00
		100		60	0,00
		100		60	0,00
					0,00
				Total	1.576,67

[d)] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado

B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e)]

Descripción	Empresa	Costes imputable
Licencia de Windows	Upper Solutions	300,00
Licencia de Desarrollo de Navision (Cesión)	Tecnocom	1.000,00
	Total	1.300,00

[e)] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,…

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	48.000
Amortización	1.577
Subcontratación de tareas	0
Costes de funcionamiento	1.300
Costes Indirectos	10.175
Total	61.052

El presupuesto total de este proyecto asciende a la cantidad de 61.052€

Leganés, 15 de Agosto de 2011

El ingeniero junior

Fdo: Miguel Ángel Carbonero Jiménez